W3School 前端教程合集

wizardforcel

Published with GitBook

目錄

W3School 前端教程合集

来源：菜鸟教程

整理：飞龙

感谢菜鸟教程站长的翻译和奉献。

W3School HTML 教程

来源：HTML教程

整理：飞龙

HTML 基础

HTML 简介

实例

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

什么是 HTML？

HTML 是用来描述网页的一种语言。

- HTML 指的是超文本标记语言 (**H**yper **T**ext **M**arkup **L**anguage)
- HTML 不是一种编程语言，而是一种标记语言 (markup language)
- 标记语言是一套标记标签 (markup tag)
- HTML 使用标记标签来描述网页

HTML 标签

HTML 标记标签通常被称为 HTML 标签 (HTML tag)。

- HTML 标签是由尖括号包围的关键词，比如 <html>
- HTML 标签通常是成对出现的，比如 和
- 标签对中的第一个标签是开始标签，第二个标签是结束标签
- 开始和结束标签也被称为开放标签和闭合标签

HTML 文档 = 网页

- HTML 文档描述网页
- HTML 文档包含 *HTML* 标签和纯文本
- HTML 文档也被称为网页

Web 浏览器的作用是读取 HTML 文档，并以网页的形式显示出它们。浏览器不会显示 HTML 标签，而是使用标签来解释页面的内容：

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

例子解释

- <html> 与 </html> 之间的文本描述网页
- <body> 与 </body> 之间的文本是可见的页面内容
- <h1> 与 </h1> 之间的文本被显示为标题
- <p> 与 </p> 之间的文本被显示为段落

基本的 HTML 标签 - 四个实例

本章通过实例向您演示最常用的 **HTML** 标签。

提示：不要担心本章中您还没有学过的例子，您将在下面的章节中学到它们。

提示：学习 HTML 最好的方式就是边学边做实验。我们为您准备了很好的 HTML 编辑器。使用这个编辑器，您可以任意编辑一段 HTML 源码，然后单击 TIY 按钮来查看结果。

HTML 标题

HTML 标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

实例

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

HTML 段落

HTML 段落是通过 <p> 标签进行定义的。

实例

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML 链接

HTML 链接是通过 <a> 标签进行定义的。

实例

```
<a href="http://www.w3school.com.cn">This is a link</a>
```

注释：在 href 属性中指定链接的地址。

（您将在本教程稍后的章节中学习更多有关属性的知识）。

HTML 图像

HTML 图像是通过 <img> 标签进行定义的。

实例

```
<img src="w3school.jpg" width="104" height="142" />
```

注释：图像的名称和尺寸是以属性的形式提供的。

HTML 元素

HTML 文档是由 **HTML** 元素定义的。

HTML 元素

HTML 元素指的是从开始标签（start tag）到结束标签（end tag）的所有代码。

开始标签	元素内容	结束标签
`<p>`	This is a paragraph	`</p>`
`<a href="default.htm" >`	This is a link	`</a>`
` `		

注释：开始标签常被称为开放标签（opening tag），结束标签常称为闭合标签（closing tag）。

HTML 元素语法

- HTML 元素以开始标签起始
- HTML 元素以结束标签终止
- 元素的内容是开始标签与结束标签之间的内容
- 某些 HTML 元素具有空内容（*empty content*）
- 空元素在开始标签中进行关闭（以开始标签的结束而结束）
- 大多数 HTML 元素可拥有属性

提示：您将在本教程的下一章中学习更多有关属性的内容。

嵌套的 HTML 元素

大多数 HTML 元素可以嵌套（可以包含其他 HTML 元素）。

HTML 文档由嵌套的 HTML 元素构成。

HTML 文档实例

```
<html>

<body>
<p>This is my first paragraph.</p>
</body>

</html>
```

上面的例子包含三个 HTML 元素。

HTML 实例解释

<p> 元素：

```
<p>This is my first paragraph.</p>
```

这个 <p> 元素定义了 HTML 文档中的一个段落。

这个元素拥有一个开始标签 <p>，以及一个结束标签 </p>。

元素内容是：This is my first paragraph。

<body> 元素：

```
<body>
<p>This is my first paragraph.</p>
</body>
```

<body> 元素定义了 HTML 文档的主体。

这个元素拥有一个开始标签 <body>，以及一个结束标签 </body>。

元素内容是另一个 HTML 元素（p 元素）。

<html> 元素：

```
<html>

<body>
<p>This is my first paragraph.</p>
</body>

</html>
```

<html> 元素定义了整个 HTML 文档。

这个元素拥有一个开始标签 <html>，以及一个结束标签 </html>。

元素内容是另一个 HTML 元素（body 元素）。

不要忘记结束标签

即使您忘记了使用结束标签，大多数浏览器也会正确地显示 HTML：

```
<p>This is a paragraph
<p>This is a paragraph
```

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生不可预料的结果或错误。

注释：未来的 HTML 版本不允许省略结束标签。

空的 HTML 元素

没有内容的 HTML 元素被称为空元素。空元素是在开始标签中关闭的。

 就是没有关闭标签的空元素（
 标签定义换行）。

在 XHTML、XML 以及未来版本的 HTML 中，所有元素都必须被关闭。

在开始标签中添加斜杠，比如
，是关闭空元素的正确方法，HTML、XHTML 和 XML 都接受这种方式。

即使
 在所有浏览器中都是有效的，但使用
 其实是更长远的保障。

HTML 提示：使用小写标签

HTML 标签对大小写不敏感：<P> 等同于 <p>。许多网站都使用大写的 HTML 标签。

W3School 使用的是小写标签，因为万维网联盟（W3C）在 HTML 4 中推荐使用小写，而在未来 (X)HTML 版本中强制使用小写。

HTML 属性

属性为 **HTML** 元素提供附加信息。

HTML 属性

HTML 标签可以拥有属性。属性提供了有关 HTML 元素的更多的信息。

属性总是以名称/值对的形式出现，比如：*name="value"*。

属性总是在 HTML 元素的开始标签中规定。

属性实例

HTML 链接由 <a> 标签定义。链接的地址在 href 属性中指定：

```
<a href="http://www.w3school.com.cn">This is a link</a>
```

更多 HTML 属性实例

属性例子 1:

<h1> 定义标题的开始。

<h1 align="center"> 拥有关于对齐方式的附加信息。

TIY：居中排列标题

属性例子 2:

<body> 定义 HTML 文档的主体。

<body bgcolor="yellow"> 拥有关于背景颜色的附加信息。

TIY：背景颜色

属性例子 3:

<table> 定义 HTML 表格。（您将在稍后的章节学习到更多有关 HTML 表格的内容）

<table border="1"> 拥有关于表格边框的附加信息。

HTML 提示：使用小写属性

属性和属性值对大小写不敏感。

不过，万维网联盟在其 HTML 4 推荐标准中推荐小写的属性/属性值。

而新版本的 (X)HTML 要求使用小写属性。

始终为属性值加引号

属性值应该始终被包括在引号内。双引号是最常用的，不过使用单引号也没有问题。

在某些个别的情况下，比如属性值本身就含有双引号，那么您必须使用单引号，例如：

```
name='Bill "HelloWorld" Gates'
```

HTML 属性参考手册

我们的完整的 HTML 参考手册提供了每个 HTML 元素可使用的合法属性的完整列表：

完整的 HTML 参考手册

下面列出了适用于大多数 HTML 元素的属性：

属性	值	描述
class	*classname*	规定元素的类名（classname）
id	*id*	规定元素的唯一 id
style	*style_definition*	规定元素的行内样式（inline style）
title	*text*	规定元素的额外信息（可在工具提示中显示）

如需更多关于标准属性的信息，请访问：

HTML 标准属性参考手册

HTML 标题

在 **HTML** 文档中，标题很重要。

HTML 标题

标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

<h1> 定义最大的标题。<h6> 定义最小的标题。

实例

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

注释：浏览器会自动地在标题的前后添加空行。

注释：默认情况下，HTML 会自动地在块级元素前后添加一个额外的空行，比如段落、标题元素前后。

标题很重要

请确保将 HTML heading 标签只用于标题。不要仅仅是为了产生粗体或大号的文本而使用标题。

搜索引擎使用标题为您的网页的结构和内容编制索引。

因为用户可以通过标题来快速浏览您的网页，所以用标题来呈现文档结构是很重要的。

应该将 h1 用作主标题（最重要的），其后是 h2（次重要的），再其次是 h3，以此类推。

HTML 水平线

<hr /> 标签在 HTML 页面中创建水平线。

hr 元素可用于分隔内容。

实例

```
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
```

提示：使用水平线 (<hr> 标签) 来分隔文章中的小节是一个办法（但并不是唯一的办法）。

HTML 注释

可以将注释插入 HTML 代码中，这样可以提高其可读性，使代码更易被人理解。浏览器会忽略注释，也不会显示它们。

注释是这样写的：

实例

```
<!-- This is a comment -->
```

注释：开始括号之后（左边的括号）需要紧跟一个叹号，结束括号之前（右边的括号）不需要。

提示：合理地使用注释可以对未来的代码编辑工作产生帮助。

HTML 提示 - 如何查看源代码

您一定曾经在看到某个网页时惊叹道 “WOW! 这是如何实现的？”

如果您想找到其中的奥秘，只需要单击右键，然后选择“查看源文件”（IE）或“查看页面源代码”（Firefox），其他浏览器的做法也是类似的。这么做会打开一个包含页面 HTML 代码的窗口。

来自本页的实例

标题

隐藏的注释

水平线

HTML 标签参考手册

W3School 的标签参考手册提供了有关这些标题及其属性的更多信息。

您将在本教程下面的章节中学到更多有关 HTML 标签和属性的知识。

标签	描述
`<html>`	定义 HTML 文档。
`<body>`	定义文档的主体。
`<h1>` to `<h6>`	定义 HTML 标题
`<hr>`	定义水平线。
`<!-->`	定义注释。

HTML 段落

可以把 **HTML** 文档分割为若干段落。

HTML 段落

段落是通过 <p> 标签定义的。

实例

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

注释：浏览器会自动地在段落的前后添加空行。（<p> 是块级元素）

提示：使用空的段落标记 <p></p> 去插入一个空行是个坏习惯。用
 标签代替它！（但是不要用
 标签去创建列表。不要着急，您将在稍后的篇幅学习到 HTML 列表。）

不要忘记结束标签

即使忘了使用结束标签，大多数浏览器也会正确地将 HTML 显示出来：

实例

```
<p>This is a paragraph
<p>This is another paragraph
```

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生意想不到的结果和错误。

注释：在未来的 HTML 版本中，不允许省略结束标签。

提示：通过结束标签来关闭 HTML 是一种经得起未来考验的 HTML 编写方法。清楚地标记某个元素在何处开始，并在何处结束，不论对您还是对浏览器来说，都会使代码更容易理解。

HTML 折行

如果您希望在不产生一个新段落的情况下进行换行（新行），请使用
 标签：

```
<p>This is<br />a para<br />graph with line breaks</p>
```


 元素是一个空的 HTML 元素。由于关闭标签没有任何意义，因此它没有结束标签。

 还是

您也许发现
 与
 很相似。

在 XHTML、XML 以及未来的 HTML 版本中，不允许使用没有结束标签（闭合标签）的 HTML 元素。

即使
 在所有浏览器中的显示都没有问题，使用
 也是更长远的保障。

HTML 输出 - 有用的提示

我们无法确定 HTML 被显示的确切效果。屏幕的大小，以及对窗口的调整都可能导致不同的结果。

对于 HTML，您无法通过在 HTML 代码中添加额外的空格或换行来改变输出的效果。

当显示页面时，浏览器会移除源代码中多余的空格和空行。所有连续的空格或空行都会被算作一个空格。需要注意的是，HTML 代码中的所有连续的空行（换行）也被显示为一个空格。

（这个例子演示了一些 HTML 格式化方面的问题）

来自本页的实例

HTML 段落

换行

在 HTML 代码中的排版一首唐诗

更多实例

更多段落

HTML 标签参考手册

W3School 的标签参考手册提供了有关 HTML 元素及其属性的更多信息。

标签	描述
`<p>`	定义段落。
` `	插入单个折行（换行）。

HTML 文本格式化

HTML 可定义很多供格式化输出的元素，比如粗体和斜体字。

下面有很多例子，您可以亲自试试：

HTML 文本格式化实例

文本格式化

```
&lt;html&gt;

&lt;body&gt;

&lt;b&gt;This text is bold&lt;/b&gt;

&lt;br /&gt;

&lt;strong&gt;This text is strong&lt;/strong&gt;

&lt;br /&gt;

&lt;big&gt;This text is big&lt;/big&gt;

&lt;br /&gt;

&lt;em&gt;This text is emphasized&lt;/em&gt;

&lt;br /&gt;

&lt;i&gt;This text is italic&lt;/i&gt;

&lt;br /&gt;

&lt;small&gt;This text is small&lt;/small&gt;

&lt;br /&gt;

This text contains
&lt;sub&gt;subscript&lt;/sub&gt;

&lt;br /&gt;

This text contains
&lt;sup&gt;superscript&lt;/sup&gt;

&lt;/body&gt;
&lt;/html&gt;
```

预格式文本

```
&lt;html&gt;

&lt;body&gt;

&lt;pre&gt;
这是
预格式文本。
它保留了      空格
和换行。
&lt;/pre&gt;

&lt;p&gt;pre 标签很适合显示计算机代码：&lt;/p&gt;

&lt;pre&gt;
for i = 1 to 10
     print i
next i
&lt;/pre&gt;

&lt;/body&gt;
&lt;/html&gt;
```

“计算机输出”标签

```
&lt;html&gt;

&lt;body&gt;

&lt;code&gt;Computer code&lt;/code&gt;
&lt;br /&gt;
&lt;kbd&gt;Keyboard input&lt;/kbd&gt;
&lt;br /&gt;
&lt;tt&gt;Teletype text&lt;/tt&gt;
&lt;br /&gt;
&lt;samp&gt;Sample text&lt;/samp&gt;
&lt;br /&gt;
&lt;var&gt;Computer variable&lt;/var&gt;
&lt;br /&gt;

&lt;p&gt;
&lt;b&gt;注释：&lt;/b&gt;这些标签常用于显示计算机/编程代码。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

地址

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;address&gt;
Written by &lt;a href="mailto:webmaster@example.com"&gt;Donald Duck
Visit us at:&lt;br&gt;
Example.com&lt;br&gt;
Box 564, Disneyland&lt;br&gt;
USA
&lt;/address&gt;

&lt;/body&gt;
&lt;/html&gt;
```

缩写和首字母缩写

```
&lt;html&gt;

&lt;body&gt;

&lt;abbr title="etcetera"&gt;etc.&lt;/abbr&gt;
&lt;br /&gt;
&lt;acronym title="World Wide Web"&gt;WWW&lt;/acronym&gt;

&lt;p&gt;在某些浏览器中，当您把鼠标移至缩略词语上时，title 可用于展示表达的完

&lt;p&gt;仅对于 IE 5 中的 acronym 元素有效。&lt;/p&gt;

&lt;p&gt;对于 Netscape 6.2 中的 abbr 和 acronym 元素都有效。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

文字方向

```
&lt;html&gt;

&lt;body&gt;

&lt;abbr title="etcetera"&gt;etc.&lt;/abbr&gt;
&lt;br /&gt;
&lt;acronym title="World Wide Web"&gt;WWW&lt;/acronym&gt;

&lt;p&gt;在某些浏览器中，当您把鼠标移至缩略词语上时，title 可用于展示表达的完

&lt;p&gt;仅对于 IE 5 中的 acronym 元素有效。&lt;/p&gt;

&lt;p&gt;对于 Netscape 6.2 中的 abbr 和 acronym 元素都有效。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

块引用

```
&lt;html&gt;

&lt;body&gt;

这是长的引用：
&lt;blockquote&gt;
这是长的引用。这是长的引用。这是长的引用。这是长的引用。这是长的引用。这是长的引
&lt;/blockquote&gt;

这是短的引用：
&lt;q&gt;
这是短的引用。
&lt;/q&gt;

&lt;p&gt;
使用 blockquote 元素的话，浏览器会插入换行和外边距，而 q 元素不会有任何特殊的
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

删除字效果和插入字效果

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;一打有 &lt;del&gt;二十&lt;/del&gt; &lt;ins&gt;十二&lt;/ins&g

&lt;p&gt;大多数浏览器会改写为删除文本和下划线文本。&lt;/p&gt;

&lt;p&gt;一些老式的浏览器会把删除文本和下划线文本显示为普通文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

如何查看 HTML 源码

您是否有过这样的经历，当你看到一个很棒的站点，你会很想知道开发人员是如何将它实现的？

你有没有看过一些网页，并且想知道它是如何做出来的呢？

要揭示一个网站的技术秘密，其实很简单。单击浏览器的“查看”菜单，选择“查看源文件”即可。随后你会看到一个弹出的窗口，窗口内就是实际的 HTML 代码。

文本格式化标签

标签	描述
<b>	定义粗体文本。
<big>	定义大号字。
<em>	定义着重文字。
<i>	定义斜体字。
<small>	定义小号字。
<strong>	定义加重语气。
<sub>	定义下标字。
<sup>	定义上标字。
<ins>	定义插入字。
<del>	定义删除字。
<s>	不赞成使用。使用 <del> 代替。
<strike>	不赞成使用。使用 <del> 代替。
<u>	不赞成使用。使用样式（style）代替。

“计算机输出”标签

标签	描述
<code>	定义计算机代码。
<kbd>	定义键盘码。
<samp>	定义计算机代码样本。
<tt>	定义打字机代码。
<var>	定义变量。
<pre>	定义预格式文本。
<listing>	不赞成使用。使用 <pre> 代替。
<plaintext>	不赞成使用。使用 <pre> 代替。
<xmp>	不赞成使用。使用 <pre> 代替。

引用、引用和术语定义

标签	描述
`<abbr>`	定义缩写。
`<acronym>`	定义首字母缩写。
`<address>`	定义地址。
`<bdo>`	定义文字方向。
`<blockquote>`	定义长的引用。
`<q>`	定义短的引用语。
`<cite>`	定义引用、引证。
`<dfn>`	定义一个定义项目。

延伸阅读：

改变文本的外观和含义

HTML 编辑器

使用 Notepad 或 TextEdit 来编写 HTML

可以使用专业的 HTML 编辑器来编辑 HTML：

- Adobe Dreamweaver
- Microsoft Expression Web
- CoffeeCup HTML Editor

不过，我们同时推荐使用文本编辑器来学习 HTML，比如 Notepad (PC) 或 TextEdit (Mac)。我们相信，使用一款简单的文本编辑器是学习 HTML 的好方法。

通过记事本，依照以下四步来创建您的第一张网页。

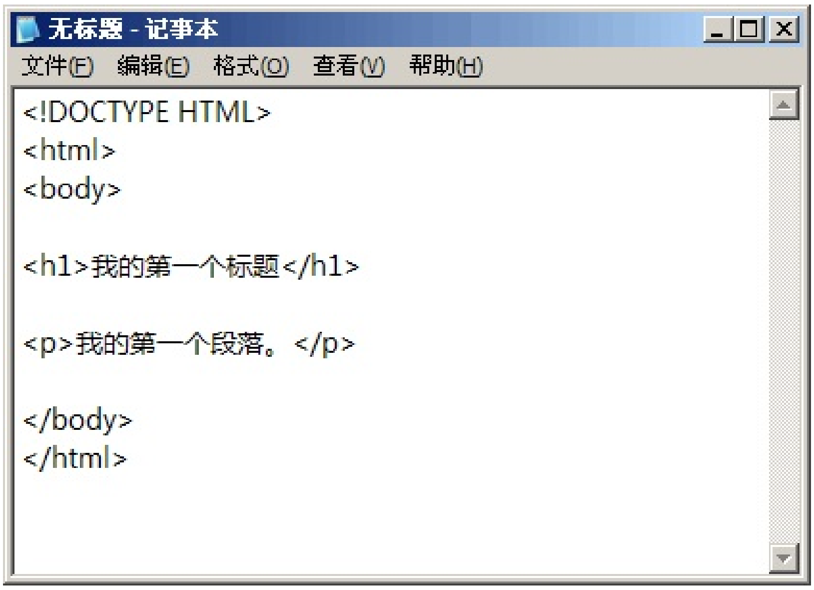
步骤一：启动记事本

如何启动记事本：

开始
所有程序
附件
记事本

步骤二：用记事本来编辑 HTML

在记事本中键入 HTML 代码：



步骤三：保存 HTML

在记事本的文件菜单选择“另存为”。

当您保存 HTML 文件时，既可以使用 .htm 也可以使用 .html 扩展名。两者没有区别，完全根据您的喜好。

在一个容易记忆的文件夹中保存这个文件，比如 w3school。

步骤四：在浏览器中运行这个 HTML 文件

启动您的浏览器，然后选择“文件”菜单的“打开文件”命令，或者直接在文件夹中双击您的 HTML 文件。

结果应该类似这样：



HTML CSS

通过使用 **HTML4.0**，所有的格式化代码均可移出 **HTML** 文档，然后移入一个独立的样式表。

实例

HTML中的样式

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
h1 {color: red}
p {color: blue}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h1&gt;header 1&lt;/h1&gt;
&lt;p&gt;A paragraph.&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

没有下划线的链接

```
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Content-Language" content="zh-cn" /&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;a href="/example/html/lastpage.html" style="text-decoration:nor
这是一个链接！
&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

链接到一个外部样式表

```
&lt;html&gt;

&lt;head&gt;
&lt;link rel="stylesheet" type="text/css" href="/html/csstest1.css"
&lt;/head&gt;

&lt;body&gt;
&lt;h1&gt;我通过外部样式表进行格式化。&lt;/h1&gt;
&lt;p&gt;我也一样！&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

如何使用样式

当浏览器读到一个样式表，它就会按照这个样式表来对文档进行格式化。有以下三种方式来插入样式表：

外部样式表

当样式需要被应用到很多页面的时候，外部样式表将是理想的选择。使用外部样式表，你就可以通过更改一个文件来改变整个站点的外观。

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

内部样式表

当单个文件需要特别样式时，就可以使用内部样式表。你可以在 head 部分通过 <style> 标签定义内部样式表。

```
<head>

<style type="text/css">
body {background-color: red}
p {margin-left: 20px}
</style>
</head>
```

内联样式

当特殊的样式需要应用到个别元素时，就可以使用内联样式。 使用内联样式的方法是在相关的标签中使用样式属性。样式属性可以包含任何 CSS 属性。以下实例显示出如何改变段落的颜色和左外边距。

```
<p style="color: red; margin-left: 20px">
This is a paragraph
</p>
```

访问我们的 CSS 教程，学习更多有关样式的知识。

标签	描述
`<style>`	定义样式定义。
`<link>`	定义资源引用。
`<div>`	定义文档中的节或区域（块级）。
`<span>`	定义文档中的行内的小块或区域。
`<font>`	规定文本的字体、字体尺寸、字体颜色。不赞成使用。请使用样式。
`<basefont>`	定义基准字体。不赞成使用。请使用样式。
`<center>`	对文本进行水平居中。不赞成使用。请使用样式。

HTML 链接

HTML 使用超级链接与网络上的另一个文档相连。

几乎可以在所有的网页中找到链接。点击链接可以从一张页面跳转到另一张页面。

实例

创建超级链接

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
&lt;a href="/index.html"&gt;本文本&lt;/a&gt; 是一个指向本网站中的一个页

&lt;p&gt;&lt;a href="http://www.microsoft.com/"&gt;本文本&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

将图像作为链接

```
&lt;html&gt;

&lt;body&gt;
&lt;p&gt;
您也可以使用图像来作链接：
&lt;a href="/example/html/lastpage.html"&gt;
&lt;img border="0" src="/i/eg_buttonnext.gif" /&gt;
&lt;/a&gt;
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例）

HTML 超链接（链接）

超链接可以是一个字，一个词，或者一组词，也可以是一幅图像，您可以点击这些内容来跳转到新的文档或者当前文档中的某个部分。

当您把鼠标指针移动到网页中的某个链接上时，箭头会变为一只小手。

我们通过使用 <a> 标签在 HTML 中创建链接。

有两种使用 <a> 标签的方式：

1. 通过使用 href 属性 - 创建指向另一个文档的链接
2. 通过使用 name 属性 - 创建文档内的书签

延伸阅读：什么是超文本？

HTML 链接语法

链接的 HTML 代码很简单。它类似这样：

```
<a href="url">Link text</a>
```

href 属性规定链接的目标。

开始标签和结束标签之间的文字被作为超级链接来显示。

实例

```
<a href="http://www.w3school.com.cn/">Visit W3School</a>
```

上面这行代码显示为：Visit W3School

点击这个超链接会把用户带到 W3School 的首页。

提示："链接文本" 不必一定是文本。图片或其他 HTML 元素都可以成为链接。

HTML 链接 - target 属性

使用 Target 属性，你可以定义被链接的文档在何处显示。

下面的这行会在新窗口打开文档：

```
<a href="http://www.w3school.com.cn/" target="_blank">Visit W3Schoo
```

HTML 链接 - name 属性

name 属性规定锚（anchor）的名称。

您可以使用 name 属性创建 HTML 页面中的书签。

书签不会以任何特殊方式显示，它对读者是不可见的。

当使用命名锚（named anchors）时，我们可以创建直接跳至该命名锚（比如页面中某个小节）的链接，这样使用者就无需不停地滚动页面来寻找他们需要的信息了。

命名锚的语法：

```
<a name="_label_">锚（显示在页面上的文本）</a>
```

提示：锚的名称可以是任何你喜欢的名字。

提示：您可以使用 id 属性来替代 name 属性，命名锚同样有效。

实例

首先，我们在 HTML 文档中对锚进行命名（创建一个书签）：

```
<a name="tips">基本的注意事项 - 有用的提示</a>
```

然后，我们在同一个文档中创建指向该锚的链接：

```
<a href="#tips">有用的提示</a>
```

您也可以在其他页面中创建指向该锚的链接：

```
<a href="http://www.w3school.com.cn/html/html_links.asp#tips">有用的
```

在上面的代码中，我们将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到 tips 这个命名锚了。

具体效果：有用的提示

基本的注意事项 - 有用的提示：

注释：请始终将正斜杠添加到子文件夹。假如这样书写链接：href="http://www.w3school.com.cn/html"，就会向服务器产生两次 HTTP 请求。这是因为服务器会添加正斜杠到这个地址，然后创建一个新的请求，就像这样：href="http://www.w3school.com.cn/html/"。

提示：命名锚经常用于在大型文档开始位置上创建目录。可以为每个章节赋予一个命名锚，然后把链接到这些锚的链接放到文档的上部。如果您经常访问百度百科，您会发现其中几乎每个词条都采用这样的导航方式。

提示：假如浏览器找不到已定义的命名锚，那么就会定位到文档的顶端。不会有错误发生。

更多实例

在新的浏览器窗口打开链接

```
&lt;html&gt;

&lt;body&gt;

&lt;a href="http://www.w3school.com.cn/" target="_blank"&gt;Visit W

&lt;p&gt;如果把链接的 target 属性设置为 "_blank"，该链接会在新窗口中打开。&

&lt;/body&gt;

&lt;/html&gt;
```

链接到同一个页面的不同位置

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
&lt;a href="#C4"&gt;查看 Chapter 4。&lt;/a&gt;
&lt;/p&gt;

&lt;h2&gt;Chapter 1&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 2&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 3&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;&lt;a name="C4"&gt;Chapter 4&lt;/a&gt;&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 5&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;
```

```
&lt;h2&gt;Chapter 6&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 7&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 8&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 9&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 10&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 11&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 12&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 13&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 14&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 15&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 16&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 17&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

跳出框架

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;被锁在框架中了吗？&lt;/p&gt;

&lt;a href="/index.html"
target="_top"&gt;请点击这里！&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建电子邮件链接

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
这是邮件链接：
&lt;a href="mailto:someone@microsoft.com?subject=Hello%20again"&gt
&lt;/p&gt;

&lt;p&gt;
&lt;b&gt;注意：&lt;/b&gt;应该使用 %20 来替换单词之间的空格，这样浏览器就可
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建电子邮件链接 2

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
这是另一个 mailto 链接：
&lt;a href="mailto:someone@microsoft.com?cc=someoneelse@microsoft.c
&lt;/p&gt;

&lt;p&gt;
&lt;b&gt;注意：&lt;/b&gt;应该使用 %20 来替换单词之间的空格，这样浏览器就可以
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

HTML 链接标签

标签	描述
<a>	定义锚。

延伸阅读

什么是超文本？

HTML 图像

通过使用 **HTML**，可以在文档中显示图像。

实例

插入图像

```
&lt;!DOCTYPE HTML&gt;
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
一幅图像：
&lt;img src="/i/eg_mouse.jpg" width="128" height="128" /&gt;
&lt;/p&gt;

&lt;p&gt;
一幅动画图像：
&lt;img src="/i/eg_cute.gif" width="50" height="50" /&gt;
&lt;/p&gt;

&lt;p&gt;请注意，插入动画图像的语法与插入普通图像的语法没有区别。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

从不同的位置插入图片

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
来自另一个文件夹的图像：
&lt;img src="/i/ct_netscape.jpg" /&gt;
&lt;/p&gt;

&lt;p&gt;
来自 W3School.com.cn 的图像：
&lt;img src="img/w3school_logo_white.gif" /&gt;
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

图像标签（<img>）和源属性（Src）

在 HTML 中，图像由 <img> 标签定义。

<img> 是空标签，意思是说，它只包含属性，并且没有闭合标签。

要在页面上显示图像，你需要使用源属性（src）。src 指 "source"。源属性的值是图像的 URL 地址。

定义图像的语法是：

```
<img src="url" />
```

URL 指存储图像的位置。如果名为 "boat.gif" 的图像位于 www.w3school.com.cn 的 images 目录中，那么其 URL 为 http://www.w3school.com.cn/images/boat.gif。

浏览器将图像显示在文档中图像标签出现的地方。如果你将图像标签置于两个段落之间，那么浏览器会首先显示第一个段落，然后显示图片，最后显示第二段。

替换文本属性（Alt）

alt 属性用来为图像定义一串预备的可替换的文本。替换文本属性的值是用户定义的。

```
<img src="boat.gif" alt="Big Boat">
```

在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息。此时，浏览器将显示这个替代性的文本而不是图像。为页面上的图像都加上替换文本属性是个好习惯，这样有助于更好的显示信息，并且对于那些使用纯文本浏览器的人来说是非常有用的。

基本的注意事项 - 有用的提示：

假如某个 HTML 文件包含十个图像，那么为了正确显示这个页面，需要加载 11 个文件。加载图片是需要时间的，所以我们的建议是：慎用图片。

更多实例

背景图片

```
&lt;html&gt;

&lt;body background="/i/eg_background.jpg"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

排列图片

```
&lt;html&gt;

&lt;body&gt;

&lt;h2&gt;未设置对齐方式的图像：&lt;/h2&gt;

&lt;p&gt;图像 &lt;img src ="/i/eg_cute.gif"&gt; 在文本中&lt;/p&gt;

&lt;h2&gt;已设置对齐方式的图像：&lt;/h2&gt;

&lt;p&gt;图像 &lt;img src="/i/eg_cute.gif" align="bottom"&gt; 在文本

&lt;p&gt;图像 &lt;img src ="/i/eg_cute.gif" align="middle"&gt; 在文

&lt;p&gt;图像 &lt;img src ="/i/eg_cute.gif" align="top"&gt; 在文本中&

&lt;p&gt;请注意，bottom 对齐方式是默认的对齐方式。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

浮动图像

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
&lt;img src ="/i/eg_cute.gif" align ="left"&gt;
带有图像的一个段落。图像的 align 属性设置为 "left"。图像将浮动到文本的左侧。
&lt;/p&gt;

&lt;p&gt;
&lt;img src ="/i/eg_cute.gif" align ="right"&gt;
带有图像的一个段落。图像的 align 属性设置为 "right"。图像将浮动到文本的右侧。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

调整图像尺寸

```
&lt;html&gt;

&lt;body&gt;

&lt;img src="/i/eg_mouse.jpg" width="50" height="50"&gt;

&lt;br /&gt;

&lt;img src="/i/eg_mouse.jpg" width="100" height="100"&gt;

&lt;br /&gt;

&lt;img src="/i/eg_mouse.jpg" width="200" height="200"&gt;

&lt;p&gt;通过改变 img 标签的 "height" 和 "width" 属性的值，您可以放大或缩

&lt;/body&gt;
&lt;/html&gt;
```

为图片显示替换文本

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;仅支持文本的浏览器无法显示图像，仅仅能够显示在图像的 "alt" 属性中指

&lt;p&gt;请注意，如果您把鼠标指针移动到图像上，大多数浏览器会显示 "alt" 文本。

&lt;img src="/i/eg_goleft.gif" alt="向左转" /&gt;

&lt;p&gt;如果无法显示图像，将显示 "alt" 属性中的文本：&lt;/p&gt;

&lt;img src="/i/eg_goleft123.gif" alt="向左转" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

制作图像链接

```
&lt;html&gt;

&lt;body&gt;
&lt;p&gt;
您也可以把图像作为链接来使用：
&lt;a href="/example/html/lastpage.html"&gt;
&lt;img border="0" src="/i/eg_buttonnext.gif" /&gt;
&lt;/a&gt;
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建图像映射

```
&lt;html&gt;
&lt;body&gt;

&lt;p&gt;请点击图像上的星球，把它们放大。&lt;/p&gt;

&lt;img
src="/i/eg_planets.jpg"
border="0" usemap="#planetmap"
alt="Planets" /&gt;

&lt;map name="planetmap" id="planetmap"&gt;

&lt;area
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
target ="_blank"
alt="Venus" /&gt;

&lt;area
shape="circle"
coords="129,161,10"
href ="/example/html/mercur.html"
target ="_blank"
alt="Mercury" /&gt;

&lt;area
shape="rect"
coords="0,0,110,260"
href ="/example/html/sun.html"
target ="_blank"
alt="Sun" /&gt;

&lt;/map&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;img 元素中的 "usemap" 属性引用 map 元

&lt;/body&gt;
&lt;/html&gt;
```

把图像转换为图像映射

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;请把鼠标移动到图像上，看一下状态栏的坐标如何变化。&lt;/p&gt;

&lt;a href="/example/html/html_ismap.html"&gt;
&lt;img src="/i/eg_planets.jpg" ismap /&gt;
&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

图像标签

标签	描述
<img>	定义图像。
<map>	定义图像地图。
<area>	定义图像地图中的可点击区域。

</table>

HTML 表格

你可以使用 **HTML** 创建表格。

实例

表格

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;每个表格由 table 标签开始。&lt;/p&gt;
&lt;p&gt;每个表格行由 tr 标签开始。&lt;/p&gt;
&lt;p&gt;每个表格数据由 td 标签开始。&lt;/p&gt;

&lt;h4&gt;一列：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;一行三列：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;两行三列：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格边框

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;带有普通的边框：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有粗的边框：&lt;/h4&gt;
&lt;table border="8"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有很粗的边框：&lt;/h4&gt;
&lt;table border="15"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

表格

表格由 <table> 标签来定义。每个表格均有若干行（由 <tr> 标签定义），每行被分割为若干单元格（由 <td> 标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器显示如下：

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

表格和边框属性

如果不定义边框属性，表格将不显示边框。有时这很有用，但是大多数时候，我们希望显示边框。

使用边框属性来显示一个带有边框的表格：

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

表格的表头

表格的表头使用 <th> 标签进行定义。

大多数浏览器会把表头显示为粗体居中的文本：

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器显示如下：

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

表格中的空单元格

在一些浏览器中，没有内容的表格单元显示得不太好。如果某个单元格是空的（没有内容），浏览器可能无法显示出这个单元格的边框。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td></td>
<td>row 2, cell 2</td>
</tr>
</table>
```

浏览器可能会这样显示：

row 1, cell 1	row 1, cell 2
	row 2, cell 2

注意：这个空的单元格的边框没有被显示出来。为了避免这种情况，在空单元格中添加一个空格占位符，就可以将边框显示出来。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td> </td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器中显示如下：

row 1, cell 1	row 1, cell 2
row 2, cell 2	

更多实例

没有边框的表格

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;这个表格没有边框：&lt;/h4&gt;
&lt;table&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;这个表格也没有边框：&lt;/h4&gt;
&lt;table border="0"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格中的表头(Heading)

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;表头：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;th&gt;电话&lt;/th&gt;
  &lt;th&gt;电话&lt;/th&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;垂直的表头：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;th&gt;电话&lt;/th&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;th&gt;电话&lt;/th&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

空单元格

```
&lt;html&gt;

&lt;body&gt;

&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;Some text&lt;/td&gt;
  &lt;td&gt;Some text&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;&lt;/td&gt;
  &lt;td&gt;Some text&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;正如您看到的，其中一个单元没有边框。这是因为它是空的。在该单元中插入-

&lt;p&gt;我们的技巧是在单元中插入一个 no-breaking 空格。&lt;/p&gt;

&lt;p&gt;no-breaking 空格是一个字符实体。如果您不清楚什么是字符实体，请阅读

&lt;p&gt;no-breaking 空格由和号开始 ("&")，然后是字符"nbsp"，并以分号结尾

&lt;/body&gt;
&lt;/html&gt;
```

带有标题的表格

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;这个表格有一个标题，以及粗边框：&lt;/h4&gt;

&lt;table border="6"&gt;
&lt;caption&gt;我的标题&lt;/caption&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
```

跨行或跨列的表格单元格

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;横跨两列的单元格：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;th colspan="2"&gt;电话&lt;/th&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;横跨两行的单元格：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;th rowspan="2"&gt;电话&lt;/th&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格内的标签

```
&lt;html&gt;

&lt;body&gt;

&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;
   &lt;p&gt;这是一个段落。&lt;/p&gt;
   &lt;p&gt;这是另一个段落。&lt;/p&gt;
  &lt;/td&gt;
  &lt;td&gt;这个单元包含一个表格：
   &lt;table border="1"&gt;
   &lt;tr&gt;
     &lt;td&gt;A&lt;/td&gt;
     &lt;td&gt;B&lt;/td&gt;
   &lt;/tr&gt;
   &lt;tr&gt;
     &lt;td&gt;C&lt;/td&gt;
     &lt;td&gt;D&lt;/td&gt;
   &lt;/tr&gt;
   &lt;/table&gt;
  &lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;这个单元包含一个列表：
   &lt;ul&gt;
    &lt;li&gt;苹果&lt;/li&gt;
    &lt;li&gt;香蕉&lt;/li&gt;
    &lt;li&gt;菠萝&lt;/li&gt;
   &lt;/ul&gt;
  &lt;/td&gt;
  &lt;td&gt;HELLO&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

单元格边距(Cell padding)

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;没有 cellpadding：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有 cellpadding：&lt;/h4&gt;
&lt;table border="1"
cellpadding="10"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

单元格间距(Cell spacing)

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;没有 cellspacing：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有 cellspacing：&lt;/h4&gt;
&lt;table border="1"
cellspacing="10"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

向表格添加背景颜色或背景图像

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;背景颜色：&lt;/h4&gt;
&lt;table border="1"
bgcolor="red"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;背景图像：&lt;/h4&gt;
&lt;table border="1"
background="/i/eg_bg_07.gif"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

向表格单元添加背景颜色或者背景图像

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;单元背景：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td bgcolor="red"&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td
  background="/i/eg_bg_07.gif"&gt;
  Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

在表格单元中排列内容

```
&lt;html&gt;

&lt;body&gt;

&lt;table width="400" border="1"&gt;
 &lt;tr&gt;
  &lt;th align="left"&gt;消费项目....&lt;/th&gt;
  &lt;th align="right"&gt;一月&lt;/th&gt;
  &lt;th align="right"&gt;二月&lt;/th&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;td align="left"&gt;衣服&lt;/td&gt;
  &lt;td align="right"&gt;$241.10&lt;/td&gt;
  &lt;td align="right"&gt;$50.20&lt;/td&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;td align="left"&gt;化妆品&lt;/td&gt;
  &lt;td align="right"&gt;$30.00&lt;/td&gt;
  &lt;td align="right"&gt;$44.45&lt;/td&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;td align="left"&gt;食物&lt;/td&gt;
  &lt;td align="right"&gt;$730.40&lt;/td&gt;
  &lt;td align="right"&gt;$650.00&lt;/td&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;th align="left"&gt;总计&lt;/th&gt;
  &lt;th align="right"&gt;$1001.50&lt;/th&gt;
  &lt;th align="right"&gt;$744.65&lt;/th&gt;
 &lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

框架(frame)属性

```
&lt;html&gt;
&lt;body&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;frame 属性无法在 Internet Explorer

&lt;p&gt;Table with frame="box":&lt;/p&gt;
&lt;table frame="box"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
```

```
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="above":&lt;/p&gt;
&lt;table frame="above"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="below":&lt;/p&gt;
&lt;table frame="below"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="hsides":&lt;/p&gt;
&lt;table frame="hsides"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="vsides":&lt;/p&gt;
&lt;table frame="vsides"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格标签

表格	描述
<table>	定义表格
<caption>	定义表格标题。
<th>	定义表格的表头。
<tr>	定义表格的行。
<td>	定义表格单元。
<thead>	定义表格的页眉。
<tbody>	定义表格的主体。
<tfoot>	定义表格的页脚。
<col>	定义用于表格列的属性。
<colgroup>	定义表格列的组。

HTML 列表

HTML 支持有序、无序和定义列表

实例

无序列表

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;一个无序列表：&lt;/h4&gt;
&lt;ul&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;茶&lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

有序列表

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;ol&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
  &lt;li&gt;茶&lt;/li&gt;
&lt;/ol&gt;

&lt;ol start="50"&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
  &lt;li&gt;茶&lt;/li&gt;
&lt;/ol&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

无序列表

无序列表是一个项目的列表，此列项目使用粗体圆点（典型的小黑圆圈）进行标记。

无序列表始于 <ul> 标签。每个列表项始于 。

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

浏览器显示如下：

- Coffee
- Milk

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

有序列表

同样，有序列表也是一列项目，列表项目使用数字进行标记。

有序列表始于 <ol> 标签。每个列表项始于 <li> 标签。

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

浏览器显示如下：

1. Coffee
2. Milk

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

定义列表

自定义列表不仅仅是一列项目，而是项目及其注释的组合。

自定义列表以 <dl> 标签开始。每个自定义列表项以 <dt> 开始。每个自定义列表项的定义以 <dd> 开始。

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

浏览器显示如下：

Coffee

Milk

定义列表的列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

更多实例

不同类型的无序列表

```
&lt;html&gt;
&lt;body&gt;

&lt;h4&gt;Disc 项目符号列表：&lt;/h4&gt;
&lt;ul type="disc"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ul&gt;

&lt;h4&gt;Circle 项目符号列表：&lt;/h4&gt;
&lt;ul type="circle"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ul&gt;

&lt;h4&gt;Square 项目符号列表：&lt;/h4&gt;
&lt;ul type="square"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

不同类型的有序列表

```
&lt;html&gt;
&lt;body&gt;

&lt;h4&gt;数字列表：&lt;/h4&gt;
&lt;ol&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;字母列表：&lt;/h4&gt;
&lt;ol type="A"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;小写字母列表：&lt;/h4&gt;
&lt;ol type="a"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;罗马字母列表：&lt;/h4&gt;
&lt;ol type="I"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;小写罗马字母列表：&lt;/h4&gt;
&lt;ol type="i"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;/body&gt;
&lt;/html&gt;
```

嵌套列表

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;一个嵌套列表：&lt;/h4&gt;
&lt;ul&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;茶
    &lt;ul&gt;
    &lt;li&gt;红茶&lt;/li&gt;
    &lt;li&gt;绿茶&lt;/li&gt;
    &lt;/ul&gt;
  &lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

嵌套列表 2

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;一个嵌套列表：&lt;/h4&gt;
&lt;ul&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;茶
    &lt;ul&gt;
    &lt;li&gt;红茶&lt;/li&gt;
    &lt;li&gt;绿茶
      &lt;ul&gt;
      &lt;li&gt;中国茶&lt;/li&gt;
      &lt;li&gt;非洲茶&lt;/li&gt;
      &lt;/ul&gt;
    &lt;/li&gt;
    &lt;/ul&gt;
  &lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

定义列表

```
&lt;html&gt;

&lt;body&gt;

&lt;h2&gt;一个定义列表：&lt;/h2&gt;

&lt;dl&gt;
   &lt;dt&gt;计算机&lt;/dt&gt;
   &lt;dd&gt;用来计算的仪器 ... ...&lt;/dd&gt;
   &lt;dt&gt;显示器&lt;/dt&gt;
   &lt;dd&gt;以视觉方式显示信息的装置 ... ...&lt;/dd&gt;
&lt;/dl&gt;

&lt;/body&gt;
&lt;/html&gt;
```

列表标签

标签	描述
	定义有序列表。
	定义无序列表。
	定义列表项。
<dl>	定义定义列表。
<dt>	定义定义项目。
<dd>	定义定义的描述。
<dir>	已废弃。使用 代替它。
<menu>	已废弃。使用 代替它。

HTML <div> 和 <span>

可以通过 **<div>** 和 **<span>** 将 **HTML** 元素组合起来。

HTML 块元素

大多数 HTML 元素被定义为块级元素或内联元素。

编者注："块级元素"译为 block level element，"内联元素"译为 inline element。

块级元素在浏览器显示时，通常会以新行来开始（和结束）。

例子：<h1>, <p>, <ul>, <table>

HTML 内联元素

内联元素在显示时通常不会以新行开始。

例子：<b>, <td>, <a>, <img>

HTML <div> 元素

HTML <div> 元素是块级元素，它是可用于组合其他 HTML 元素的容器。

<div> 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。

如果与 CSS 一同使用，<div> 元素可用于对大的内容块设置样式属性。

<div> 元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式方法。使用 <table> 元素进行文档布局不是表格的正确用法。<table> 元素的作用是显示表格化的数据。

HTML <span> 元素

HTML <span> 元素是内联元素，可用作文本的容器。

<span> 元素也没有特定的含义。

当与 CSS 一同使用时，<span> 元素可用于为部分文本设置样式属性。

HTML 分组标签

标签	描述
<div>	定义文档中的分区或节（division/section）。
<span>	定义 span，用来组合文档中的行内元素。

HTML 布局

网页布局对改善网站的外观非常重要。

请慎重设计您的网页布局。

亲自试一试 - 实例

使用 <div> 元素的网页布局

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
div#container{width:500px}
div#header {background-color:#99bbbb;}
div#menu {background-color:#ffff99;height:200px;width:150px;float:l
div#content {background-color:#EEEEEE;height:200px;width:350px;floa
div#footer {background-color:#99bbbb;clear:both;text-align:center;}
h1 {margin-bottom:0;}
h2 {margin-bottom:0;font-size:18px;}
ul {margin:0;}
li {list-style:none;}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;div id="container"&gt;

&lt;div id="header"&gt;
&lt;h1&gt;Main Title of Web Page&lt;/h1&gt;
&lt;/div&gt;

&lt;div id="menu"&gt;
&lt;h2&gt;Menu&lt;/h2&gt;
&lt;ul&gt;
&lt;li&gt;HTML&lt;/li&gt;
&lt;li&gt;CSS&lt;/li&gt;
&lt;li&gt;JavaScript&lt;/li&gt;
&lt;/ul&gt;
&lt;/div&gt;

&lt;div id="content"&gt;Content goes here&lt;/div&gt;

&lt;div id="footer"&gt;Copyright W3School.com.cn&lt;/div&gt;

&lt;/div&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用 <table> 元素的网页布局

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;table width="500" border="0"&gt;
&lt;tr&gt;
&lt;td colspan="2" style="background-color:#99bbbb;"&gt;
&lt;h1&gt;Main Title of Web Page&lt;/h1&gt;
&lt;/td&gt;
&lt;/tr&gt;

&lt;tr valign="top"&gt;
&lt;td style="background-color:#ffff99;width:100px;text-align:top;"
&lt;b&gt;Menu&lt;/b&gt;&lt;br /&gt;
HTML&lt;br /&gt;
CSS&lt;br /&gt;
JavaScript
&lt;/td&gt;
&lt;td style="background-color:#EEEEEE;height:200px;width:400px;tex
Content goes here&lt;/td&gt;
&lt;/tr&gt;

&lt;tr&gt;
&lt;td colspan="2" style="background-color:#99bbbb;text-align:cente
Copyright W3School.com.cn&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

网站布局

大多数网站会把内容安排到多个列中（就像杂志或报纸那样）。

可以使用 <div> 或者 <table> 元素来创建多列。CSS 用于对元素进行定位，或者为页面创建背景以及色彩丰富的外观。

提示：即使可以使用 HTML 表格来创建漂亮的布局，但设计表格的目的是呈现表格化数据 - 表格不是布局工具！

HTML 布局 - 使用 <div> 元素

div 元素是用于分组 HTML 元素的块级元素。

下面的例子使用五个 div 元素来创建多列布局：

实例

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
div#container{width:500px}
div#header {background-color:#99bbbb;}
div#menu {background-color:#ffff99; height:200px; width:100px; floa
div#content {background-color:#EEEEEE; height:200px; width:400px; f
div#footer {background-color:#99bbbb; clear:both; text-align:center
h1 {margin-bottom:0;}
h2 {margin-bottom:0; font-size:14px;}
ul {margin:0;}
li {list-style:none;}
</style>
</head>

<body>

<div id="container">

<div id="header">
<h1>Main Title of Web Page</h1>
</div>

<div id="menu">
<h2>Menu</h2>
<ul>
<li>HTML</li>
<li>CSS</li>
<li>JavaScript</li>
</ul>
</div>

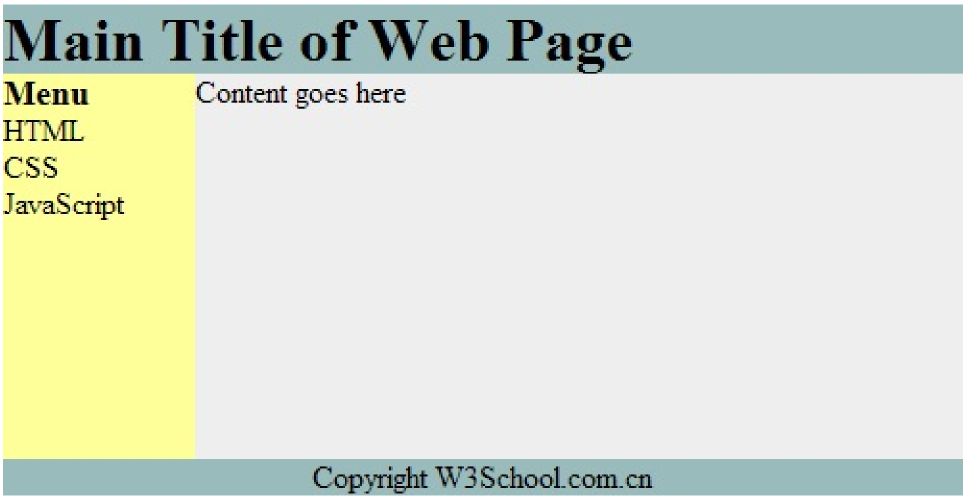
<div id="content">Content goes here</div>

<div id="footer">Copyright W3School.com.cn</div>

</div>

</body>
</html>
```

上面的 HTML 代码会产生如下结果：



HTML 布局 - 使用表格

使用 HTML <table> 标签是创建布局的一种简单的方式。

可以使用 <div> 或者 <table> 元素来创建多列。CSS 用于对元素进行定位，或者为页面创建背景以及色彩丰富的外观。

提示：即使可以使用 HTML 表格来创建漂亮的布局，但设计表格的目的是呈现表格化数据 - 表格不是布局工具！

下面的例子使用三行两列的表格 - 第一和最后一行使用 colspan 属性来横跨两列：

实例

```
<!DOCTYPE html>
<html>
<body>

<table width="500" border="0">
<tr>
<td colspan="2" style="background-color:#99bbbb;">
<h1>Main Title of Web Page</h1>
</td>
</tr>

<tr valign="top">
<td style="background-color:#ffff99;width:100px;text-align:top;">
<b>Menu</b><br />
HTML<br />
CSS<br />
JavaScript
</td>
<td style="background-color:#EEEEEE;height:200px;width:400px;text-a
Content goes here</td>
</tr>

<tr>
<td colspan="2" style="background-color:#99bbbb;text-align:center;'
Copyright W3School.com.cn</td>
</tr>
</table>

</body>
</html>
```

上面的 HTML 代码会产生以下结果：



HTML 布局 - 有用的提示

提示：使用 CSS 最大的好处是，如果把 CSS 代码存放到外部样式表中，那么站点会更易于维护。通过编辑单一的文件，就可以改变所有页面的布局。如需学习更多有关 CSS 的知识，请访问我们的 CSS 教程。

提示：由于创建高级的布局非常耗时，使用模板是一个快速的选项。通过搜索引擎可以找到很多免费的网站模板（您可以使用这些预先构建好的网站布局，并优化它们）。

HTML 布局标签

标签	描述
`<div>`	定义文档中的分区或节（division/section）。
`<span>`	定义 span，用来组合文档中的行内元素。

HTML 表单和输入

HTML 表单用于搜集不同类型的用户输入。

实例

文本域 (Text field)

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
名：
&lt;input type="text" name="firstname"&gt;
&lt;br /&gt;
姓：
&lt;input type="text" name="lastname"&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

密码域

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
用户：
&lt;input type="text" name="user"&gt;
&lt;br /&gt;
密码：
&lt;input type="password" name="password"&gt;
&lt;/form&gt;
&lt;p&gt;
请注意，当您在密码域中键入字符时，浏览器将使用项目符号来代替这些字符。
&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

表单

表单是一个包含表单元素的区域。

表单元素是允许用户在表单中（比如：文本域、下拉列表、单选框、复选框等等）输入信息的元素。

表单使用表单标签（<form>）定义。

```
<form>
...
  input 元素
...
</form>
```

输入

多数情况下被用到的表单标签是输入标签（<input>）。输入类型是由类型属性（type）定义的。大多数经常被用到的输入类型如下：

文本域（**Text Fields**）

当用户要在表单中键入字母、数字等内容时，就会用到文本域。

```
<form>
First name:
<input type="text" name="firstname" />
<br />
Last name:
<input type="text" name="lastname" />
</form>
```

浏览器显示如下：

First name: Last name:

注意，表单本身并不可见。同时，在大多数浏览器中，文本域的缺省宽度是20个字符。

单选按钮（**Radio Buttons**）

当用户从若干给定的的选择中选取其一时，就会用到单选框。

```
<form>
<input type="radio" name="sex" value="male" /> Male
<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

浏览器显示如下：

○ Male ○ Female
注意，只能从中选取其一。

复选框（Checkboxes）

当用户需要从若干给定的选择中选取一个或若干选项时，就会用到复选框。

```
<form>
<input type="checkbox" name="bike" />
I have a bike
<br />
<input type="checkbox" name="car" />
I have a car
</form>
```

浏览器显示如下：

☐ I have a bike ☐ I have a car

表单的动作属性（Action）和确认按钮

当用户单击确认按钮时，表单的内容会被传送到另一个文件。表单的动作属性定义了目的文件的文件名。由动作属性定义的这个文件通常会对接收到的输入数据进行相关的处理。

```
<form name="input" action="html_form_action.asp" method="get">
Username:
<input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

浏览器显示如下：

Username: [ ] Submit

假如您在上面的文本框内键入几个字母，然后点击确认按钮，那么输入数据会传送到 "html_form_action.asp" 的页面。该页面将显示出输入的结果。

更多实例

复选框

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
我喜欢自行车：
&lt;input type="checkbox" name="Bike"&gt;
&lt;br /&gt;
我喜欢汽车：
&lt;input type="checkbox" name="Car"&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

单选按钮

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
男性：
&lt;input type="radio" checked="checked" name="Sex" value="male" /&
&lt;br /&gt;
女性：
&lt;input type="radio" name="Sex" value="female" /&gt;
&lt;/form&gt;

&lt;p&gt;当用户点击一个单选按钮时，该按钮会变为选中状态，其他所有按钮会变为非

&lt;/body&gt;
&lt;/html&gt;
```

简单的下拉列表

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
&lt;select name="cars"&gt;
&lt;option value="volvo"&gt;Volvo&lt;/option&gt;
&lt;option value="saab"&gt;Saab&lt;/option&gt;
&lt;option value="fiat"&gt;Fiat&lt;/option&gt;
&lt;option value="audi"&gt;Audi&lt;/option&gt;
&lt;/select&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

另一个下拉列表

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
&lt;select name="cars"&gt;
&lt;option value="volvo"&gt;Volvo&lt;/option&gt;
&lt;option value="saab"&gt;Saab&lt;/option&gt;
&lt;option value="fiat" selected="selected"&gt;Fiat&lt;/option&gt;
&lt;option value="audi"&gt;Audi&lt;/option&gt;
&lt;/select&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

文本域(Textarea)

```
&lt;html&gt;
&lt;body&gt;

&lt;p&gt;
This example cannot be edited
because our editor uses a textarea
for input,
and your browser does not allow
a textarea inside a textarea.
&lt;/p&gt;

&lt;textarea rows="10" cols="30"&gt;
The cat was playing in the garden.
```

创建按钮

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
&lt;input type="button" value="Hello world!"&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

Fieldset around data

```
&lt;!DOCTYPE HTML&gt;
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
  &lt;fieldset&gt;
    &lt;legend&gt;健康信息&lt;/legend&gt;
    身高：&lt;input type="text" /&gt;
    体重：&lt;input type="text" /&gt;
  &lt;/fieldset&gt;
&lt;/form&gt;

&lt;p&gt;如果表单周围没有边框，说明您的浏览器太老了。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表单实例

带有输入框和确认按钮的表单

```
&lt;html&gt;
&lt;body&gt;

&lt;form action="/example/html/form_action.asp" method="get"&gt;
  &lt;p&gt;First name: &lt;input type="text" name="fname" /&gt;&lt;
  &lt;p&gt;Last name: &lt;input type="text" name="lname" /&gt;&lt;
  &lt;input type="submit" value="Submit" /&gt;
&lt;/form&gt;

&lt;p&gt;请单击确认按钮，输入会发送到服务器上名为 "form_action.asp" 的页面

&lt;/body&gt;
&lt;/html&gt;
```

带有复选框的表单

```
&lt;html&gt;

&lt;body&gt;

&lt;form name="input" action="/html/html_form_action.asp" method="
I have a bike:
&lt;input type="checkbox" name="vehicle" value="Bike" checked="chec
&lt;br /&gt;
I have a car:
&lt;input type="checkbox" name="vehicle" value="Car" /&gt;
&lt;br /&gt;
I have an airplane:
&lt;input type="checkbox" name="vehicle" value="Airplane" /&gt;
&lt;br /&gt;&lt;br /&gt;
&lt;input type="submit" value="Submit" /&gt;
&lt;/form&gt;

&lt;p&gt;如果您点击 "Submit" 按钮，您将把输入传送到名为 html_form_action

&lt;/body&gt;
&lt;/html&gt;
```

带有单选按钮的表单

```
&lt;html&gt;

&lt;body&gt;

&lt;form name="input" action="/html/html_form_action.asp" method="g
Male:
&lt;input type="radio" name="Sex" value="Male" checked="checked"&gt
&lt;br /&gt;
Female:
&lt;input type="radio" name="Sex" value="Female"&gt;
&lt;br /&gt;
&lt;input type ="submit" value ="Submit"&gt;
&lt;/form&gt;

&lt;p&gt;如果您点击 "Submit" 按钮，您将把输入传送到名为 html_form_action

&lt;/body&gt;
&lt;/html&gt;
```

从表单发送电子邮件

```
&lt;html&gt;

&lt;body&gt;
&lt;form action="MAILTO:someone@w3school.com.cn" method="post" enct

&lt;h3&gt;这个表单会把电子邮件发送到 W3School。&lt;/h3&gt;
姓名：&lt;br /&gt;
&lt;input type="text" name="name" value="yourname" size="20"&gt;
&lt;br /&gt;
电邮：&lt;br /&gt;
&lt;input type="text" name="mail" value="yourmail" size="20"&gt;
&lt;br /&gt;
内容：&lt;br /&gt;
&lt;input type="text" name="comment" value="yourcomment" size="40"&
&lt;br /&gt;&lt;br /&gt;
&lt;input type="submit" value="发送"&gt;
&lt;input type="reset" value="重置"&gt;

&lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;
```

表单标签

标签	描述
<form>	定义供用户输入的表单
<input>	定义输入域
<textarea>	定义文本域 (一个多行的输入控件)
<label>	定义一个控制的标签
<fieldset>	定义域
<legend>	定义域的标题
<select>	定义一个选择列表
<optgroup>	定义选项组
<option>	定义下拉列表中的选项
<button>	定义一个按钮
<isindex>	已废弃。由 <input> 代替。

HTML 框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。

实例

垂直框架

```
&lt;html&gt;

&lt;frameset cols="25%,50%,25%"&gt;

  &lt;frame src="/example/html/frame_a.html"&gt;
  &lt;frame src="/example/html/frame_b.html"&gt;
  &lt;frame src="/example/html/frame_c.html"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

水平框架

```
&lt;html&gt;

&lt;frameset rows="25%,50%,25%"&gt;

  &lt;frame src="/example/html/frame_a.html"&gt;
  &lt;frame src="/example/html/frame_b.html"&gt;
  &lt;frame src="/example/html/frame_c.html"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。每份HTML文档称为一个框架，并且每个框架都独立于其他的框架。

使用框架的坏处：

- 开发人员必须同时跟踪更多的HTML文档

- 很难打印整张页面

框架结构标签（<frameset>）

编者注：frameset 标签也被某些文章和书籍译为框架集。

框架标签（Frame）

Frame 标签定义了放置在每个框架中的 HTML 文档。

在下面的这个例子中，我们设置了一个两列的框架集。第一列被设置为占据浏览器窗口的 25%。第二列被设置为占据浏览器窗口的 75%。HTML 文档 "frame_a.htm" 被置于第一个列中，而 HTML 文档 "frame_b.htm" 被置于第二个列中：

```
<frameset cols="25%,75%">
   <frame src="frame_a.htm">
   <frame src="frame_b.htm">
</frameset>
```

基本的注意事项 - 有用的提示：

假如一个框架有可见边框，用户可以拖动边框来改变它的大小。为了避免这种情况发生，可以在 <frame> 标签中加入：noresize="noresize"。

为不支持框架的浏览器添加 <noframes> 标签。

重要提示：不能将 <body></body> 标签与 <frameset></frameset> 标签同时使用！不过，假如你添加包含一段文本的 <noframes> 标签，就必须将这段文字嵌套于 <body></body> 标签内。（在下面的第一个实例中，可以查看它是如何实现的。）

更多实例

如何使用 <noframes> 标签

```
&lt;html&gt;

&lt;frameset cols="25%,50%,25%"&gt;
  &lt;frame src="/example/html/frame_a.html"&gt;
  &lt;frame src="/example/html/frame_b.html"&gt;
  &lt;frame src="/example/html/frame_c.html"&gt;

&lt;noframes&gt;
&lt;body&gt;您的浏览器无法处理框架！&lt;/body&gt;
&lt;/noframes&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

混合框架结构

```
&lt;html&gt;

&lt;frameset rows="50%,50%"&gt;

&lt;frame src="/example/html/frame_a.html"&gt;

&lt;frameset cols="25%,75%"&gt;
&lt;frame src="/example/html/frame_b.html"&gt;
&lt;frame src="/example/html/frame_c.html"&gt;
&lt;/frameset&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

含有 noresize="noresize" 属性的框架结构

```
&lt;html&gt;

&lt;frameset cols="50%,*,25%"&gt;
  &lt;frame src="/example/html/frame_a.html" noresize="noresize" /&
  &lt;frame src="/example/html/frame_b.html" /&gt;
  &lt;frame src="/example/html/frame_c.html" /&gt;
&lt;/frameset&gt;

&lt;/html&gt;
```

导航框架

```
&lt;html&gt;

&lt;frameset cols="120,*"&gt;

  &lt;frame src="/example/html/html_contents.html"&gt;
  &lt;frame src="/example/html/frame_a.html" name="showframe"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

内联框架

```
&lt;html&gt;

&lt;body&gt;

&lt;iframe src="/i/eg_landscape.jpg"&gt;&lt;/iframe&gt;

&lt;p&gt;一些老的浏览器不支持 iframe。&lt;/p&gt;
&lt;p&gt;如果得不到支持，iframe 是不可见的。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

跳转至框架内的一个指定的节

```
&lt;html&gt;

&lt;frameset cols="20%,80%"&gt;

 &lt;frame src="/example/html/frame_a.html"&gt;
 &lt;frame src="/example/html/link.html#C10"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

使用框架导航跳转至指定的节

HTML Iframe

iframe 用于在网页内显示网页。

```
<iframe src="/html/index.asp" height="300px" width="99%" style="mar
```

添加 iframe 的语法

```
<iframe src="_URL_"></iframe>
```

URL 指向隔离页面的位置。

Iframe - 设置高度和宽度

height 和 width 属性用于规定 iframe 的高度和宽度。

属性值的默认单位是像素，但也可以用百分比来设定（比如 "80%"）。

实例

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Iframe - 删除边框

frameborder 属性规定是否显示 iframe 周围的边框。

设置属性值为 "0" 就可以移除边框：

实例

```
<iframe src="demo_iframe.htm" frameborder="0"></iframe>
```

使用 iframe 作为链接的目标

iframe 可用作链接的目标（target）。

链接的 target 属性必须引用 iframe 的 name 属性：

实例

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.w3school.com.cn" target="iframe_a">W3School
```

HTML iframe 标签

标签	描述
`<iframe>`	定义内联的子窗口（框架）

HTML 背景

好的背景使站点看上去特别棒。

实例

搭配良好的背景和颜色

```
&lt;html&gt;

&lt;body bgcolor="#d0d0d0"&gt;

&lt;p&gt;
这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。
&lt;/p&gt;

&lt;p&gt;
这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

搭配得不好的背景和颜色

```
&lt;html&gt;

&lt;body bgcolor="#ffffff" text="yellow"&gt;

&lt;p&gt;
这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。
&lt;/p&gt;

&lt;p&gt;
这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

背景（Backgrounds）

<body> 拥有两个配置背景的标签。背景可以是颜色或者图像。

背景颜色（Bgcolor）

背景颜色属性将背景设置为某种颜色。属性值可以是十六进制数、RGB 值或颜色名。

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

以上的代码均将背景颜色设置为黑色。

背景（Background）

背景属性将背景设置为图像。属性值为图像的URL。如果图像尺寸小于浏览器窗口，那么图像将在整个浏览器窗口进行复制。

```
<body background="clouds.gif">
<body background="http://www.w3school.com.cn/clouds.gif">
```

URL可以是相对地址，如第一行代码。也可以使绝对地址，如第二行代码。

提示：如果你打算使用背景图片，你需要紧记一下几点：

- 背景图像是否增加了页面的加载时间。小贴士：图像文件不应超过 10k。
- 背景图像是否与页面中的其他图象搭配良好。
- 背景图像是否与页面中的文字颜色搭配良好。
- 图像在页面中平铺后，看上去还可以吗？
- 对文字的注意力被背景图像喧宾夺主了吗？

基本的注意事项 - 有用的提示：

<body> 标签中的背景颜色（bgcolor）、背景（background）和文本（text）属性在最新的 HTML 标准（HTML4 和 XHTML）中已被废弃。W3C 在他们的推荐标准中已删除这些属性。

应该使用层叠样式表（CSS）来定义 HTML 元素的布局和显示属性。

更多实例

可用性强的背景图像

```
&lt;html&gt;

&lt;body background="/i/eg_bg_06.gif"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

可用性强的背景图像 2

```
&lt;html&gt;

&lt;body background="/i/eg_bg_04.gif"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

可用性差的背景图像

```
&lt;html&gt;

&lt;body background="/i/eg_bg_01.gif"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

HTML 颜色

颜色由红色、绿色、蓝色混合而成。

颜色值

颜色由一个十六进制符号来定义，这个符号由红色、绿色和蓝色的值组成（RGB）。

每种颜色的最小值是0（十六进制：#00）。最大值是255（十六进制：#FF）。

这个表格给出了由三种颜色混合而成的具体效果：

Color HEX	Color RGB
#000000	rgb(0,0,0)
#FF0000	rgb(255,0,0)
#00FF00	rgb(0,255,0)
#0000FF	rgb(0,0,255)
#FFFF00	rgb(255,255,0)
#00FFFF	rgb(0,255,255)
#FF00FF	rgb(255,0,255)
#C0C0C0	rgb(192,192,192)
#FFFFFF	rgb(255,255,255)

颜色名

大多数的浏览器都支持颜色名集合。

提示：仅仅有 16 种颜色名被 W3C 的 HTML4.0 标准所支持。它们是：aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow。

如果需要使用其它的颜色，需要使用十六进制的颜色值。

Color HEX	Color Name
#F0F8FF	AliceBlue
#FAEBD7	AntiqueWhite
#7FFFD4	Aquamarine
#000000	Black
#0000FF	Blue
#8A2BE2	BlueViolet
#A52A2A	Brown

Web安全色

数年以前，当大多数计算机仅支持 256 种颜色的时候，一系列 216 种 Web 安全色作为 Web 标准被建议使用。其中的原因是，微软和 Mac 操作系统使用了 40 种不同的保留的固定系统颜色（双方大约各使用 20 种）。

我们不确定如今这么做的意义有多大，因为越来越多的计算机有能力处理数百万种颜色，不过做选择还是你自己。

216 跨平台色

最初，216 跨平台 web 安全色被用来确保：当计算机使用 256 色调色板时，所有的计算机能够正确地显示所有的颜色。

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF

660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

HTML 颜色名

本页提供了被大多数浏览器支持的颜色名。

提示：仅有 16 种颜色名被 W3C 的 HTML 4.0 标准支持，它们是：aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white、yellow。

如果使用其它颜色的话，就应该使用十六进制的颜色值。

颜色名列表

单击一个颜色名或者 16 进制值，就可以查看与不同文字颜色搭配的背景颜色。

颜色名	十六进制颜色值
AliceBlue	#F0F8FF
AntiqueWhite	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBCD
Blue	#0000FF
BlueViolet	#8A2BE2
Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50
CornflowerBlue	#6495ED

Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
Darkorange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray	#696969
DodgerBlue	#1E90FF
Feldspar	#D19275
FireBrick	#B22222
FloralWhite	#FFFAF0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC

GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateBlue	#8470FF
LightSlateGray	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0

Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE

PaleVioletRed	#D87093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8
Tomato	#FF6347
Turquoise	#40E0D0
Violet	#EE82EE

VioletRed	#D02090
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00
YellowGreen	#9ACD32

HTML 4.01 快速参考

来自 **W3School** 的 **HTML** 快速参考。可以打印它，以备日常使用。

HTML Basic Document

```
<html>
<head>
<title>Document name goes here</title>
</head>
<body>
Visible text goes here
</body>
</html>
```

Text Elements

```
<p>This is a paragraph</p>
<br> (line break)
<hr> (horizontal rule)
<pre>This text is preformatted</pre>
```

Logical Styles

```
<em>This text is emphasized</em>
<strong>This text is strong</strong>
<code>This is some computer code</code>
```

Physical Styles

```
<b>This text is bold</b>
<i>This text is italic</i>
```

Links, Anchors, and Image Elements

```
<a href="http://www.example.com/">This is a Link</a>
<a href="http://www.example.com/"><img src="URL"
alt="Alternate Text"></a>
<a href="mailto:webmaster@example.com">Send e-mail</a>A named ancho
<a name="tips">Useful Tips Section</a>
<a href="#tips">Jump to the Useful Tips Section</a>
```

Unordered list

```
<ul>
<li>First item</li>
<li>Next item</li>
</ul>
```

Ordered list

```
<ol>
<li>First item</li>
<li>Next item</li>
</ol>
```

Definition list

```
<dl>
<dt>First term</dt>
<dd>Definition</dd>
<dt>Next term</dt>
<dd>Definition</dd>
</dl>
```

Tables

```
<table border="1">
<tr>
  <th>someheader</th>
  <th>someheader</th>
</tr>
<tr>
  <td>sometext</td>
  <td>sometext</td>
</tr>
</table>
```

Frames

```
<frameset cols="25%,75%">
  <frame src="page1.htm">
  <frame src="page2.htm">
</frameset>
```

Forms

```
<form action="http://www.example.com/test.asp" method="post/get">
<input type="text" name="lastname"
value="Nixon" size="30" maxlength="50">
<input type="password">
<input type="checkbox" checked="checked">
<input type="radio" checked="checked">
<input type="submit">
<input type="reset">
<input type="hidden">
<select>
<option>Apples
<option selected>Bananas
<option>Cherries
</select>
<textarea name="Comment" rows="60"
cols="20"></textarea>
</form>
```

Entities

```
&lt; is the same as <
&gt; is the same as >
&#169; is the same as ?
```

Other Elements

```
<!-- This is a comment -->
<blockquote>
Text quoted from some source.
</blockquote>
<address>
Address 1<br>
Address 2<br>
City<br>
</address>
```

Source : http://www.w3school.com.cn/html/html_quick.asp

HTML 高级

HTML <!DOCTYPE>

<!DOCTYPE> 声明帮助浏览器正确地显示网页。

<!DOCTYPE> 声明

Web 世界中存在许多不同的文档。只有了解文档的类型，浏览器才能正确地显示文档。

HTML 也有多个不同的版本，只有完全明白页面中使用的确切 HTML 版本，浏览器才能完全正确地显示出 HTML 页面。这就是 <!DOCTYPE> 的用处。

<!DOCTYPE> 不是 HTML 标签。它为浏览器提供一项信息（声明），即 HTML 是用什么版本编写的。

提示：W3School 即将升级为最新的 HTML5 文档类型。

实例

带有 HTML5 DOCTYPE 的 HTML 文档：

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

HTML 版本

从 Web 诞生早期至今，已经发展出多个 HTML 版本：

版本	年份
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML 1.0	2000
HTML5	2012
XHTML5	2013

常用的声明

HTML5

```
<!DOCTYPE html>
```

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

如需完整的文档类型声明列表，请访问我们的 DOCTYPE 参考手册。

HTML 头部元素

亲自试一试 - 实例

文档的标题

```
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Content-Language" content="zh-cn" /&gt;

&lt;title&gt;标题不会显示在文档区&lt;/title&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;这段文本会显示出来。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

所有链接一个目标

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Content-Language" content="zh-cn" /&gt;

&lt;base target="_blank" /&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
&lt;a href="http://www.w3school.com.cn" target="_blank"&gt;这个连接&
&lt;/p&gt;

&lt;p&gt;
&lt;a href="http://www.w3school.com.cn"&gt;这个连接&lt;/a&gt; 也将在
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

文档描述

```
&lt;html&gt;

&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23

&lt;meta name="author"
content="w3school.com.cn"&gt;

&lt;meta name="revised"
content="David Yang,8/1/07"&gt;

&lt;meta name="generator"
content="Dreamweaver 8.0en"&gt;

&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;本文档的 meta 属性标识了创作者和编辑软件。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

文档关键词

```
&lt;html&gt;

&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23

&lt;meta name="description"
content="HTML examples"&gt;

&lt;meta name="keywords"
content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript"&gt;

&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;本文档的 meta 属性描述了该文档和它的关键词。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

重定向用户

```
&lt;!DOCTYPE HTML&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Refresh" content="5;url=http://www.w3school.co
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;
对不起。我们已经搬家了。您的 URL 是 &lt;a href="http://www.w3school.com
&lt;/p&gt;

&lt;p&gt;您将在 5 秒内被重定向到新的地址。&lt;/p&gt;

&lt;p&gt;如果超过 5 秒后您仍然看到本消息，请点击上面的链接。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

HTML <head> 元素

<head> 元素是所有头部元素的容器。<head> 内的元素可包含脚本，指示浏览器在何处可以找到样式表，提供元信息，等等。

以下标签都可以添加到 head 部分：<title>、<base>、<link>、<meta>、<script> 以及 <style>。

HTML <title> 元素

<title> 标签定义文档的标题。

title 元素在所有 HTML/XHTML 文档中都是必需的。

title 元素能够：

- 定义浏览器工具栏中的标题
- 提供页面被添加到收藏夹时显示的标题
- 显示在搜索引擎结果中的页面标题

一个简化的 HTML 文档：

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

HTML <base> 元素

<base> 标签为页面上的所有链接规定默认地址或默认目标（target）：

```
<head>
<base href="http://www.w3school.com.cn/images/" />
<base target="_blank" />
</head>
```

HTML <link> 元素

<link> 标签定义文档与外部资源之间的关系。

<link> 标签最常用于连接样式表：

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

HTML <style> 元素

<style> 标签用于为 HTML 文档定义样式信息。

您可以在 style 元素内规定 HTML 元素在浏览器中呈现的样式：

```
<head>
<style type="text/css">
body {background-color:yellow}
p {color:blue}
</style>
</head>
```

HTML <meta> 元素

元数据（metadata）是关于数据的信息。

<meta> 标签提供关于 HTML 文档的元数据。元数据不会显示在页面上，但是对于机器是可读的。

典型的情况是，meta 元素被用于规定页面的描述、关键词、文档的作者、最后修改时间以及其他元数据。

<meta> 标签始终位于 head 元素中。

元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。

针对搜索引擎的关键词

一些搜索引擎会利用 meta 元素的 name 和 content 属性来索引您的页面。

下面的 meta 元素定义页面的描述：

```
<meta name="description" content="Free Web tutorials on HTML, CSS,
```

下面的 meta 元素定义页面的关键词：

```
<meta name="keywords" content="HTML, CSS, XML" />
```

name 和 content 属性的作用是描述页面的内容。

HTML <script> 元素

<script> 标签用于定义客户端脚本，比如 JavaScript。

我们会在稍后的章节讲解 script 元素。

HTML 头部元素

标签	描述
<head>	定义关于文档的信息。
<title>	定义文档标题。
<base>	定义页面上所有链接的默认地址或默认目标。
<link>	定义文档与外部资源之间的关系。
<meta>	定义关于 HTML 文档的元数据。
<script>	定义客户端脚本。
<style>	定义文档的样式信息。

HTML 脚本

JavaScript 使 **HTML** 页面具有更强的动态和交互性。。

实例

插入一段脚本

```
&lt;html&gt;

&lt;body&gt;

&lt;script type="text/javascript"&gt;
document.write("&lt;h1&gt;Hello World!&lt;/h1&gt;")
&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
```

使用 <noscript> 标签

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;script type="text/javascript"&gt;
document.write("Hello World!")
&lt;/script&gt;
&lt;noscript&gt;Sorry, your browser does not support JavaScript!&lt

&lt;p&gt;不支持 JavaScript 的浏览器将显示 noscript 元素中的文本。&lt;/p&

&lt;/body&gt;
&lt;/html&gt;
```

HTML script 元素

<script> 标签用于定义客户端脚本，比如 JavaScript。

script 元素既可包含脚本语句，也可通过 src 属性指向外部脚本文件。

必需的 type 属性规定脚本的 MIME 类型。

JavaScript 最常用于图片操作、表单验证以及内容动态更新。

下面的脚本会向浏览器输出“Hello World!”：

```
<script type="text/javascript">
document.write("Hello World!")
</script>
```

提示：如果需要学习更多有关在 HTML 中编写脚本的知识，请访问我们的 JavaScript 教程。

<noscript> 标签

<noscript> 标签提供无法使用脚本时的替代内容，比方在浏览器禁用脚本时，或浏览器不支持客户端脚本时。

noscript 元素可包含普通 HTML 页面的 body 元素中能够找到的所有元素。

只有在浏览器不支持脚本或者禁用脚本时，才会显示 noscript 元素中的内容：

```
<script type="text/javascript">
document.write("Hello World!")
</script>
<noscript>Your browser does not support JavaScript!</noscript>
```

如何应付老式的浏览器

如果浏览器压根没法识别 <script> 标签，那么 <script> 标签所包含的内容将以文本方式显示在页面上。为了避免这种情况发生，你应该将脚本隐藏在注释标签当中。那些老的浏览器（无法识别 <script> 标签的浏览器）将忽略这些注释，所以不会将标签的内容显示到页面上。而那些新的浏览器将读懂这些脚本并执行它们，即使代码被嵌套在注释标签内。

实例

JavaScript:

```
<script type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>
```

VBScript:

```
<script type="text/vbscript">
<!--
document.write("Hello World!")
'-->
</script>
```

标签	描述
`<script>`	定义客户端脚本。
`<noscript>`	为不支持客户端脚本的浏览器定义替代内容。

HTML 字符实体

HTML 中的预留字符必须被替换为字符实体。

HTML 实体

在 HTML 中，某些字符是预留的。

在 HTML 中不能使用小于号（<）和大于号（>），这是因为浏览器会误认为它们是标签。

如果希望正确地显示预留字符，我们必须在 HTML 源代码中使用字符实体（character entities）。

字符实体类似这样：

```
&_entity_name_;

或者

&#_entity_number_;
```

如需显示小于号，我们必须这样写：< 或 <

提示：使用实体名而不是数字的好处是，名称易于记忆。不过坏处是，浏览器也许并不支持所有实体名称（对实体数字的支持却很好）。

不间断空格（non-breaking space）

HTML 中的常用字符实体是不间断空格( )。

浏览器总是会截短 HTML 页面中的空格。如果您在文本中写 10 个空格，在显示该页面之前，浏览器会删除它们中的 9 个。如需在页面中增加空格的数量，您需要使用   字符实体。

HTML 实例示例

用 HTML 实体符号做实验：

HTML 中有用的字符实体

注释：实体名称对大小写敏感！

显示结果	描述	实体名称	实体编号
	空格		
<	小于号	<	<
>	大于号	>	>
&	和号	&	&
"	引号	"	"
'	撇号	' (IE不支持)	'
¢	分	¢	¢
£	镑	£	£
¥	日圆	¥	¥
€	欧元	€	€
§	小节	§	§
©	版权	©	©
®	注册商标	®	®
™	商标	™	™
×	乘号	×	×
÷	除号	÷	÷

如需完整的实体符号参考，请访问我们的 HTML 实体符号参考手册。

HTML 统一资源定位器

URL 也被称为网址。

URL 可以由单词组成，比如 “w3school.com.cn”，或者是因特网协议（IP）地址：192.168.1.253。大多数人在网上冲浪时，会键入网址的域名，因为名称比数字容易记忆。

URL - Uniform Resource Locator

当您点击 HTML 页面中的某个链接时，对应的 <a> 标签指向万维网上的一个地址。

统一资源定位器（URL）用于定位万维网上的文档（或其他数据）。

网址，比如 http://www.w3school.com.cn/html/index.asp，遵守以下的语法规则：

```
scheme://host.domain:port/path/filename
```

解释：

- scheme - 定义因特网服务的类型。最常见的类型是 http
- host - 定义域主机（http 的默认主机是 www）
- domain - 定义因特网域名，比如 w3school.com.cn
- :port - 定义主机上的端口号（http 的默认端口号是 80）
- path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。
- filename - 定义文档/资源的名称

编者注：URL 的英文全称是 Uniform Resource Locator，中文也译为“统一资源定位符”。

URL Schemes

以下是其中一些最流行的 scheme：

Scheme	访问	用于...
http	超文本传输协议	以 http:// 开头的普通网页。不加密。
https	安全超文本传输协议	安全网页。加密所有信息交换。
ftp	文件传输协议	用于将文件下载或上传至网站。
file	您计算机上的文件。	

HTML URL 字符编码

URL 编码会将字符转换为可通过因特网传输的格式。

URL - 统一资源定位器

Web 浏览器通过 URL 从 web 服务器请求页面。

URL 是网页的地址，比如 *http://www.w3school.com.cn*。

URL 编码

URL 只能使用 ASCII 字符集来通过因特网进行发送。

由于 URL 常常会包含 ASCII 集合之外的字符，URL 必须转换为有效的 ASCII 格式。

URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。

URL 不能包含空格。URL 编码通常使用 + 来替换空格。

亲自试一试

如果您点击下面的“提交”按钮，浏览器会在发送输入之前对其进行 URL 编码。服务器上的页面会显示出接收到的输入。

[   ] Submit

试着输入一些字符，然后再次点击提交按钮。

URL 编码示例

字符	**URL 编码**
€	%80
£	%A3
?	%A9
?	%AE
à	%C0
á	%C1
?	%C2
?	%C3
?	%C4
?	%C5

如需完整的 URL 编码参考，请访问我们的 URL 编码参考手册。

HTML Web Server

如果希望向世界发布您的网站，那么您必须把它存放在 **web** 服务器上。

托管自己的网站

在自己的服务器上托管网站始终是一个选项。有几点需要考虑：

硬件支出

如果要运行“真正”的网站，您不得不购买强大的服务器硬件。不要指望低价的 PC 能够应付这些工作。您还需要稳定的（一天 24 小时）高速连接。

软件支出

请记住，服务器授权通常比客户端授权更昂贵。同时请注意，服务器授权也许有用户数量限制。

人工费

不要指望低廉的人工费用。您必须安装自己的硬件和软件。您同时要处理漏洞和病毒，以确保您的服务器时刻正常地运行于一个“任何事都可能发生”的环境中。

使用因特网服务提供商（**ISP**）

从 ISP 租用服务器也很常见。

大多数小公司会把网站存放到由 ISP 提供的服务器上。其优势有以下几点：

连接速度

大多数 ISP 都拥有连接因特网的高速连接。

强大的硬件

ISP 的 web 服务器通常强大到能够由若干网站分享资源。您还要看一下 ISP 是否提供高效的负载平衡，以及必要的备份服务器。

安全性和可靠性

ISP 是网站托管方面的专家。他们应该提供 99% 以上的在线时间，最新的软件补丁，以及最好的病毒防护。

选择 ISP 时的注意事项

24 小时支持

确保 ISP 提供 24 小时支持。不要使自己置于无法解决严重问题的尴尬境地，同时还必须等待第二个工作日。如果您不希望支付长途电话费，那么免费电话服务也是必要的。

每日备份

确保 ISP 会执行每日备份的例行工作，否则您有可能损失有价值的数据。

流量

研究一下 ISP 的流量限制。如果出现由于网站受欢迎而激增的不可预期的访问量，那么您要确保不会因此支付额外费用。

带宽或内容限制

研究一下 ISP 的带宽和内容限制。如果您计划发布图片或播出视频或音频，请确保您有此权限。

E-mail 功能

请确保 ISP 支持您需要的 e-mail 功能。

数据库访问

如果您计划使用网站数据库中的数据，那么请确保您的 ISP 支持您需要的数据库访问。

在您选取一家 ISP 之前，请务必阅读 W3School 的 Web 主机教程。

HTML 媒体

HTML 多媒体

Web 上的多媒体指的是音效、音乐、视频和动画。

现代网络浏览器已支持很多多媒体格式。

什么是多媒体？

多媒体来自多种不同的格式。它可以是您听到或看到的任何内容，文字、图片、音乐、音效、录音、电影、动画等等。

在因特网上，您会经常发现嵌入网页中的多媒体元素，现代浏览器已支持多种多媒体格式。

在本教程中，您将了解到不同的多媒体格式，以及如何在您的网页中使用它们。

浏览器支持

第一款因特网浏览器只支持文本，而且即使是对文本的支持也仅限于单一字体和单一颜色。随后诞生了支持颜色、字体和文本样式的浏览器，图片支持也被加入。

不同的浏览器以不同的方式处理对音效、动画和视频的支持。某些元素能够以内联的方式处理，而某些则需要额外的插件。

您将在下面的章节学习更多有关插件的知识。

多媒体格式

多媒体元素（比如视频和音频）存储于媒体文件中。

确定媒体类型的最常用的方法是查看文件扩展名。当浏览器得到文件扩展名 .htm 或 .html 时，它会假定该文件是 HTML 页面。.xml 扩展名指示 XML 文件，而 .css 扩展名指示样式表。图片格式则通过 .gif 或 .jpg 来识别。

多媒体元素元素也拥有带有不同扩展名的文件格式，比如 .swf、.wmv、.mp3 以及 .mp4。

视频格式

MP4 格式是一种新的即将普及的因特网视频格式。HTML5 、Flash 播放器以及优酷等视频网站均支持它。

格式	文件	描述
AVI	.avi	AVI (Audio Video Interleave) 格式是由微软开发的。所有运行 Windows 的计算机都支持 AVI 格式。它是因特网上很常见的格式，但非 Windows 计算机并不总是能够播放。
WMV	.wmv	Windows Media 格式是由微软开发的。Windows Media 在因特网上很常见，但是如果未安装额外的（免费）组件，就无法播放 Windows Media 电影。一些后期的 Windows Media 电影在所有非 Windows 计算机上都无法播放，因为没有合适的播放器。
MPEG	.mpg .mpeg	MPEG (Moving Pictures Expert Group) 格式是因特网上最流行的格式。它是跨平台的，得到了所有最流行的浏览器的支持。
QuickTime	.mov	QuickTime 格式是由苹果公司开发的。QuickTime 是因特网上常见的格式，但是 QuickTime 电影不能在没有安装额外的（免费）组件的 Windows 计算机上播放。
RealVideo	.rm .ram	RealVideo 格式是由 Real Media 针对因特网开发的。该格式允许低带宽条件下（在线视频、网络电视）的视频流。由于是低带宽优先的，质量常会降低。
Flash	.swf .flv	Flash (Shockwave) 格式是由 Macromedia 开发的。Shockwave 格式需要额外的组件来播放。但是该组件会预装到 Firefox 或 IE 之类的浏览器上。
Mpeg-4	.mp4	Mpeg-4 (with H.264 video compression) 是一种针对因特网的新格式。事实上，YouTube 推荐使用 MP4。YouTube 接收多种格式，然后全部转换为 .flv 或 .mp4 以供分发。越来越多的视频发布者转到 MP4，将其作为 Flash 播放器和 HTML5 的因特网共享格式。

声音格式

格式	文件	描述
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface) 是一种针对电子音乐设备（比如合成器和声卡）的格式。MIDI 文件不含有声音，但包含可被电子产品（比如声卡）播放的数字音乐指令。点击这里播放 The Beatles。 因为 MIDI 格式仅包含指令，所以 MIDI 文件极其小巧。上面的例子只有 23k 的大小，但却能播放将近 5 分钟。MIDI 得到了广泛的平台上的大量软件的支持。大多数流行的网络浏览器都支持 MIDI。
RealAudio	.rm .ram	RealAudio 格式是由 Real Media 针对因特网开发的。该格式也支持视频。该格式允许低带宽条件下的音频流（在线音乐、网络音乐）。由于是低带宽优先的，质量常会降低。
Wave	.wav	Wave (waveform) 格式是由 IBM 和微软开发的。所有运行 Windows 的计算机和所有网络浏览器（除了 Google Chrome）都支持它。
WMA	.wma	WMA 格式 (Windows Media Audio)，质量优于 MP3，兼容大多数播放器，除了 iPod。WMA 文件可作为连续的数据流来传输，这使它对于网络电台或在线音乐很实用。
MP3	.mp3 .mpga	MP3 文件实际上是 MPEG 文件的声音部分。MPEG 格式最初是由运动图像专家组开发的。MP3 是其中最受欢迎的针对音乐的声音格式。期待未来的软件系统都支持它。

使用哪种格式？

WAVE 是因特网上最受欢迎的无压缩声音格式，所有流行的浏览器都支持它。如果您需要未经压缩的声音（音乐或演讲），那么您应该使用 WAVE 格式。

MP3 是最新的压缩录制音乐格式。MP3 这个术语已经成为数字音乐的代名词。如果您的网址从事录制音乐，那么 MP3 是一个选项。

HTML Object 元素

<object> 的作用是支持 **HTML** 助手（插件）。

HTML 助手（插件）

辅助应用程序（helper application）是可由浏览器启动的程序。辅助应用程序也称为插件。

辅助程序可用于播放音频和视频（以及其他）。辅助程序是使用 <object> 标签来加载的。

使用辅助程序播放视频和音频的一个优势是，您能够允许用户来控制部分或全部播放设置。

大多数辅助应用程序允许对音量设置和播放功能（比如后退、暂停、停止和播放）的手工（或程序的）控制。

在 HTML 中播放视频的最好方式？

如需了解在 HTML 中包含音视频的最好方法，请参阅下一章节。

使用 QuickTime 来播放 Wave 音频

实例

```
<object width="420" height="360"
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
<param name="src" value="bird.wav" />
<param name="controller" value="true" />
</object>
```

使用 QuickTime 来播放 MP4 视频

实例

```
<object width="420" height="360"
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
<param name="src" value="movie.mp4" />
<param name="controller" value="true" />
</object>
```

使用 Flash 来播放 SWF 视频

实例

```
<object width="400" height="40"
classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/
pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">
<param name="SRC" value="bookmark.swf">
<embed src="bookmark.swf" width="400" height="40"></embed>
</object>
```

使用 Windows Media Player 来播放 WMV 影片

下面的例子展示用于播放 Windows 媒体文件的推荐代码：

实例

```
<object width="100%" height="100%"
type="video/x-ms-asf" url="3d.wmv" data="3d.wmv"
classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6">
<param name="url" value="3d.wmv">
<param name="filename" value="3d.wmv">
<param name="autostart" value="1">
<param name="uiMode" value="full" />
<param name="autosize" value="1">
<param name="playcount" value="1">
<embed type="application/x-mplayer2" src="3d.wmv" width="100%"
 height="100%" autostart="true" showcontrols="true"
pluginspage="http://www.microsoft.com/Windows/MediaPlayer/"></embe
</object>
```

HTML 音频

在 **HTML** 中播放声音的方法有很多种。

问题，问题，以及解决方法

在 HTML 中播放音频并不容易！

您需要谙熟大量技巧，以确保您的音频文件在所有浏览器中（Internet Explorer, Chrome, Firefox, Safari, Opera）和所有硬件上（PC, Mac , iPad, iPhone）都能够播放。

在本章，W3School 为您总结了问题和解决方法。

使用插件

浏览器插件是一种扩展浏览器标准功能的小型计算机程序。

插件有很多用途：播放音乐、显示地图、验证银行账号，控制输入等等。

可使用 <object> 或 <embed> 标签来将插件添加到 HTML 页面。

这些标签定义资源（通常非 HTML 资源）的容器，根据类型，它们即会由浏览器显示，也会由外部插件显示。

使用 **<embed>** 元素

<embed> 标签定义外部（非 HTML）内容的容器。（这是一个 HTML5 标签，在 HTML4 中是非法的，但是所有浏览器中都有效）。

下面的代码片段能够显示嵌入网页中的 MP3 文件：

实例

```
<embed height="100" width="100" src="song.mp3" />
```

问题：

- <embed> 标签在 HTML 4 中是无效的。页面无法通过 HTML 4 验证。
- 不同的浏览器对音频格式的支持也不同。
- 如果浏览器不支持该文件格式，没有插件的话就无法播放该音频。
- 如果用户的计算机未安装插件，无法播放音频。

- 如果把该文件转换为其他格式，仍然无法在所有浏览器中播放。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

使用 **<object>** 元素

<object tag> 标签也可以定义外部（非 HTML）内容的容器。

下面的代码片段能够显示嵌入网页中的 MP3 文件：

实例

```
<object height="100" width="100" data="song.mp3"></object>
```

问题：

- 不同的浏览器对音频格式的支持也不同。
- 如果浏览器不支持该文件格式，没有插件的话就无法播放该音频。
- 如果用户的计算机未安装插件，无法播放音频。
- 如果把该文件转换为其他格式，仍然无法在所有浏览器中播放。

使用 **HTML5 <audio>** 元素

<audio> 元素是一个 HTML5 元素，在 HTML 4 中是非法的，但在所有浏览器中都有效。

实例

```
<audio controls="controls">
  <source src="song.mp3" type="audio/mp3" />
  <source src="song.ogg" type="audio/ogg" />
Your browser does not support this audio format.
</audio>
```

上面的例子使用了一个 mp3 文件，这样它在 Internet Explorer、Chrome 以及 Safari 中是有效的。

为了使这段音频在 Firefox 和 Opera 中同样有效，添加了一个 ogg 类型的文件。如果失败，会显示错误消息。

问题：

- <audio> 标签在 HTML 4 中是无效的。您的页面无法通过 HTML 4 验证。

- 您必须把音频文件转换为不同的格式。
- <audio> 元素在老式浏览器中不起作用。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

最好的 **HTML** 解决方法

实例

```
<audio controls="controls" height="100" width="100">
  <source src="song.mp3" type="audio/mp3" />
  <source src="song.ogg" type="audio/ogg" />
<embed height="100" width="100" src="song.mp3" />
</audio>
```

上面的例子使用了两个不同的音频格式。HTML5 <audio> 元素会尝试以 mp3 或 ogg 来播放音频。如果失败，代码将回退尝试 <embed> 元素。

问题：

- 您必须把音频转换为不同的格式。
- <audio> 元素无法通过 HTML 4 和 XHTML 验证。
- <embed> 元素无法通过 HTML 4 和 XHTML 验证。
- <embed> 元素无法回退来显示错误消息。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

向网站添加音频的最简单方法

向网页添加音频的最简单的方法是什么？

雅虎的媒体播放器绝对算其中之一。

使用雅虎媒体播放器是一个不同的途径。您只需简单地让雅虎来完成歌曲播放的工作就好了。

它能播放 mp3 以及一系列其他格式。通过一行简单的代码，您就可以把它添加到网页中，轻松地将 HTML 页面转变为专业的播放列表。

雅虎媒体播放器

实例

```
<a href="song.mp3">Play Sound</a>

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js
</script>
```

使用雅虎播放器是免费的。如需使用它，您需要把这段 JavaScript 插入网页底部：

```
<script type="text/javascript" src="http://mediaplayer.yahoo.com/js
```

然后只需简单地把 MP3 文件链接到您的 HTML 中，JavaScript 会自动地为每首歌创建播放按钮：

```
<a href="song1.mp3">Play Song 1</a>
<a href="song2.mp3">Play Song 2</a>
...
...
...
```

雅虎媒体播放器为您的用户提供的是一个小型的播放按钮，而不是完整的播放器。不过，当您点击该按钮，会弹出完整的播放器。

请注意，这个播放器始终停靠在窗框底部。只需点击它，就可将其滑出。

使用超链接

如果网页包含指向媒体文件的超链接，大多数浏览器会使用“辅助应用程序”来播放文件。

以下代码片段显示指向 mp3 文件的链接。如果用户点击该链接，浏览器会启动“辅助应用程序”来播放该文件：

实例

```
<a href="song.mp3">Play the sound</a>
```

内联的声音

当您在网页中包含声音，或者作为网页的组成部分时，它被称为内联声音。

如果您打算在 web 应用程序中使用内联声音，您需要意识到很多人都觉得内联声音令人恼火。同时请注意，用户可能已经关闭了浏览器中的内联声音选项。

我们最好的建议是只在用户希望听到内联声音的地方包含它们。一个正面的例子是，在用户需要听到录音并点击某个链接时，会打开页面然后播放录音。

HTML 4.01 多媒体标签

标签	描述
<applet>	不赞成。定义内嵌 applet。
<embed>	HTML4 中不赞成，HTML5 中允许。定义内嵌对象。
<object>	定义内嵌对象。
<param>	定义对象的参数。

HTML 5 多媒体标签

标签	描述
<audio>	标签定义声音，比如音乐或其他音频流。
<embed>	标签定义嵌入的内容，比如插件。

HTML 视频

在 **HTML** 中播放视频的方法有很多种。

实例

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
  <object data="movie.mp4" width="320" height="240">
    <embed src="movie.swf" width="320" height="240" />
  </object>
</video>
```

问题，问题，以及解决方法

在 HTML 中播放视频并不容易！

您需要谙熟大量技巧，以确保您的视频文件在所有浏览器中（Internet Explorer, Chrome, Firefox, Safari, Opera）和所有硬件上（PC, Mac , iPad, iPhone）都能够播放。

在本章，W3School 为您总结了问题和解决方法。

使用 <embed> 标签

<embed> 标签的作用是在 HTML 页面中嵌入多媒体元素。

下面的 HTML 代码显示嵌入网页的 Flash 视频：

实例

```
<embed src="movie.swf" height="200" width="200"/>
```

问题

- HTML4 无法识别 <embed> 标签。您的页面无法通过验证。
- 如果浏览器不支持 Flash，那么视频将无法播放
- iPad 和 iPhone 不能显示 Flash 视频。

- 如果您将视频转换为其他格式，那么它仍然不能在所有浏览器中播放。

使用 **<object>** 标签

<object> 标签的作用是在 HTML 页面中嵌入多媒体元素。

下面的 HTML 片段显示嵌入网页的一段 Flash 视频：

实例

```
<object data="movie.swf" height="200" width="200"/>
```

问题

- 如果浏览器不支持 Flash，将无法播放视频。
- iPad 和 iPhone 不能显示 Flash 视频。
- 如果您将视频转换为其他格式，那么它仍然不能在所有浏览器中播放。

使用 **<video>** 标签

<video> 是 HTML 5 中的新标签。

<video> 标签的作用是在 HTML 页面中嵌入视频元素。

以下 HTML 片段会显示一段嵌入网页的 ogg、mp4 或 webm 格式的视频：

实例

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
Your browser does not support the video tag.
</video>
```

问题

- 您必须把视频转换为很多不同的格式。
- <video> 元素在老式浏览器中无效。
- <video> 元素无法通过 HTML 4 和 XHTML 验证。

最好的 **HTML** 解决方法

HTML 5 + <object> + <embed>

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
  <object data="movie.mp4" width="320" height="240">
    <embed src="movie.swf" width="320" height="240" />
  </object>
</video>
```

上例中使用了 4 中不同的视频格式。HTML 5 <video> 元素会尝试播放以 mp4、ogg 或 webm 格式中的一种来播放视频。如果均失败，则回退到 <embed> 元素。

问题

- 您必须把视频转换为很多不同的格式
- <video> 元素无法通过 HTML 4 和 XHTML 验证。
- <embed> 元素无法通过 HTML 4 和 XHTML 验证。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

优酷解决方案

在 HTML 中显示视频的最简单的方法是使用优酷等视频网站。

如果您希望在网页中播放视频，那么您可以把视频上传到优酷等视频网站，然后在您的网页中插入 HTML 代码即可播放视频：

```
<embed src="http://player.youku.com/player.php/sid/XMzI2NTc4NTMy/v
width="480" height="400"
type="application/x-shockwave-flash">
</embed>
```

使用超链接

如果网页包含指向媒体文件的超链接，大多数浏览器会使用“辅助应用程序”来播放文件。

以下代码片段显示指向 AVI 文件的链接。如果用户点击该链接，浏览器会启动“辅助应用程序”，比如 Windows Media Player 来播放这个 AVI 文件：

实例

```
<a href="movie.swf">Play a video file</a>
```

关于内联视频的一段注释

当视频被包含在网页中时，它被称为内联视频。

如果您打算在 web 应用程序中使用内联视频，您需要意识到很多人都觉得内联视频令人恼火。

同时请注意，用户可能已经关闭了浏览器中的内联视频选项。

我们最好的建议是只在用户希望看到内联视频的地方包含它们。一个正面的例子是，在用户需要看到视频并点击某个链接时，会打开页面然后播放视频。

HTML 4.01 多媒体标签

标签	描述
<applet>	不赞成。定义内嵌 applet。
<embed>	不赞成。定义内嵌对象。（HTML5 中允许）
<object>	定义内嵌对象。
<param>	定义对象的参数。

HTML 5 多媒体标签

标签	描述
<video>	标签定义声音，比如音乐或其他音频流。
<embed>	标签定义嵌入的内容，比如插件。

HTML XHTML

XHTML 简介

XHTML 是以 XML 格式编写的 HTML。

什么是 XHTML？

- XHTML 指的是可扩展超文本标记语言
- XHTML 与 HTML 4.01 几乎是相同的
- XHTML 是更严格更纯净的 HTML 版本
- XHTML 是以 XML 应用的方式定义的 HTML
- XHTML 是 2001 年 1 月发布的 W3C 推荐标准
- XHTML 得到所有主流浏览器的支持

为什么使用 XHTML？

因特网上的很多页面包含了“糟糕”的 HTML。

如果在浏览器中查看，下面的 HTML 代码运行起来非常正常（即使它并未遵守 HTML 规则）：

```
<html>
<head>
<title>This is bad HTML</title>
<body>
<h1>Bad HTML
<p>This is a paragraph
</body>
```

XML 是一种必须正确标记且格式良好的标记语言。

如果希望学习 XML，请阅读我们的 XML 教程。

今日的科技界存在一些不同的浏览器技术。其中一些在计算机上运行，而另一些可能在移动电话或其他小型设备上运行。小型设备往往缺乏解释“糟糕”的标记语言的资源和能力。

所以 - 通过结合 XML 和 HTML 的长处，开发出了 XHTML。XHTML 是作为 XML 被重新设计的 HTML。

与 HTML 相比最重要的区别：

文档结构

- XHTML DOCTYPE 是强制性的

- <html> 中的 XML namespace 属性是强制性的
- <html>、<head>、<title> 以及 <body> 也是强制性的

元素语法

- XHTML 元素必须正确嵌套
- XHTML 元素必须始终关闭
- XHTML 元素必须小写
- XHTML 文档必须有一个根元素

属性语法

- XHTML 属性必须使用小写
- XHTML 属性值必须用引号包围
- XHTML 属性最小化也是禁止的

<!DOCTYPE ....> 是强制性的

XHTML 文档必须进行 XHTML 文档类型声明（XHTML DOCTYPE declaration）。

您可以在 W3School 的标签参考手册中找到完整的 XHTML 文档类型。

<html>、<head>、<title> 以及 <body> 元素也必须存在，并且必须使用 <html> 中的 xmlns 属性为文档规定 xml 命名空间。

下面的例子展示了带有最少的必需标签的 XHTML 文档：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>Title of document</title>
</head>

<body>
......
</body>

</html>
```

如何从 HTML 转换到 XHTML

1. 向每张页面的第一行添加 XHTML <!DOCTYPE>
2. 向每张页面的 html 元素添加 xmlns 属性

3. 把所有元素名改为小写
4. 关闭所有空元素
5. 把所有属性名改为小写
6. 为所有属性值加引号

用 W3C 验证器检验 XHTML

在下面的文本框中输入您的网址：

<input name="uri" size="60" style="margin:10px 0 0 0;" value="http://www.w3school.com.cn/html/index.asp"> <input type="submit" value="验证文件">

XHTML 测验

该测验包含 20 道问题，且没有时间限制。

本测验是非正式的，它仅仅是了解您 XHTML 知识掌握程度的一个不错的途径。

每项正确答案可获得 1 分。在测试结束后，会显示您的总分。最高分为 20 分。

开始 XHTML 测验

XHTML - 元素

XHTML 元素是以 **XML** 格式编写的 **HTML** 元素。

XHTML 元素 - 语法规则

- XHTML 元素必须正确嵌套
- XHTML 元素必须始终关闭
- XHTML 元素必须小写
- XHTML 文档必须有一个根元素

XHTML 元素必须正确嵌套

在 HTML 中，某些元素可以不正确地彼此嵌套在一起，就像这样：

```
<b><i>This text is bold and italic</b></i>
```

在 XHTML 中，所有元素必须正确地彼此嵌套，就像这样：

```
<b><i>This text is bold and italic</i></b>
```

XHTML 元素必须始终关闭

这是错误的：

```
<p>This is a paragraph
<p>This is another paragraph
```

这是正确的：

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

空元素也必须关闭

这是错误的：

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

这是正确的：

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

XHTML 元素必须小写

这是错误的：

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

这是正确的：

```
<body>
<p>This is a paragraph</p>
</body>
```

XHTML - 属性

XHTML 属性是以 **XML** 格式编写的 **HTML** 属性。

XHTML 属性 - 语法规则

- XHTML 属性必须使用小写
- XHTML 属性值必须用引号包围
- XHTML 属性最小化也是禁止的

XHTML 属性必须使用小写

这是错误的：

```
<table WIDTH="100%">
```

这是正确的：

```
<table width="100%">
```

XHTML 属性值必须用引号包围

这是错误的：

```
<table width=100%>
```

这是正确的：

```
<table width="100%">
```

禁止属性简写

这是错误的：

```
<input checked>
<input readonly>
<input disabled>
<option selected>
```

这是正确的：

```
<input checked="checked" />
<input readonly="readonly" />
<input disabled="disabled" />
<option selected="selected" />
```

CSS 基础

HTML 5 简介

HTML5 是下一代的 **HTML**。

什么是 HTML5？

HTML5 将成为 HTML、XHTML 以及 HTML DOM 的新标准。

HTML 的上一个版本诞生于 1999 年。自从那以后，Web 世界已经经历了巨变。

HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。

HTML5 是如何起步的？

HTML5 是 W3C 与 WHATWG 合作的结果。

编者注：W3C 指 World Wide Web Consortium，万维网联盟。

编者注：WHATWG 指 Web Hypertext Application Technology Working Group。

WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

为 HTML5 建立的一些规则：

- 新特性应该基于 HTML、CSS、DOM 以及 JavaScript。
- 减少对外部插件的需求（比如 Flash）
- 更优秀的错误处理
- 更多取代脚本的标记
- HTML5 应该独立于设备
- 开发进程应对公众透明

新特性

HTML5 中的一些有趣的新特性：

- 用于绘画的 canvas 元素
- 用于媒介回放的 video 和 audio 元素
- 对本地离线存储的更好的支持
- 新的特殊内容元素，比如 article、footer、header、nav、section
- 新的表单控件，比如 calendar、date、time、email、url、search

浏览器支持

最新版本的 Safari、Chrome、Firefox 以及 Opera 支持某些 HTML5 特性。Internet Explorer 9 将支持某些 HTML5 特性。

HTML 5 视频

许多时髦的网站都提供视频。**HTML5** 提供了展示视频的标准。

Web 上的视频

直到现在，仍然不存在一项旨在网页上显示视频的标准。

今天，大多数视频是通过插件（比如 Flash）来显示的。然而，并非所有浏览器都拥有同样的插件。

HTML5 规定了一种通过 video 元素来包含视频的标准方法。

视频格式

当前，video 元素支持三种视频格式：

格式	IE	Firefox	Opera	Chrome	Safari
Ogg	No	3.5+	10.5+	5.0+	No
MPEG 4	9.0+	No	No	5.0+	3.0+
WebM	No	4.0+	10.6+	6.0+	No

Ogg = 带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件

MPEG4 = 带有 H.264 视频编码和 AAC 音频编码的 MPEG 4 文件

WebM = 带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件

如何工作

如需在 HTML5 中显示视频，您所有需要的是：

```
<video src="movie.ogg" controls="controls">
</video>
```

control 属性供添加播放、暂停和音量控件。

包含宽度和高度属性也是不错的主意。

<video> 与 </video> 之间插入的内容是供不支持 video 元素的浏览器显示的：

实例

```
<video src="movie.ogg" width="320" height="240" controls="controls'
Your browser does not support the video tag.
</video>
```

上面的例子使用一个 Ogg 文件，适用于Firefox、Opera 以及 Chrome 浏览器。

要确保适用于 Safari 浏览器，视频文件必须是 MPEG4 类型。

video 元素允许多个 source 元素。source 元素可以链接不同的视频文件。浏览器将使用第一个可识别的格式：

实例

```
<video width="320" height="240" controls="controls">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

Internet Explorer

Internet Explorer 8 不支持 video 元素。在 IE 9 中，将提供对使用 MPEG4 的 video 元素的支持。

<video> 标签的属性

属性	值	描述
autoplay	autoplay	如果出现该属性，则视频在就绪后马上播放。
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮。
height	*pixels*	设置视频播放器的高度。
loop	loop	如果出现该属性，则当媒介文件完成播放后再次开始播放。
preload	preload	如果出现该属性，则视频在页面加载时进行加载，并预备播放。 如果使用 "autoplay"，则忽略该属性。
src	*url*	要播放的视频的 URL。
width	*pixels*	设置视频播放器的宽度。

相关页面

参考手册：HTML 5 <video> 标签

HTML 5 Video + DOM

HTML5 <video> - 使用 DOM 进行控制

HTML5 <video> 元素同样拥有方法、属性和事件。

其中的方法用于播放、暂停以及加载等。其中的属性（比如时长、音量等）可以被读取或设置。其中的 DOM 事件能够通知您，比方说，<video> 元素开始播放、已暂停，已停止，等等。

下例中简单的方法，向我们演示了如何使用 <video> 元素，读取并设置属性，以及如何调用方法。

实例

为视频创建简单的播放/暂停以及调整尺寸控件：

```
<!DOCTYPE html>
<html>
<body>

<div style="text-align:center;">
  <button onclick="playPause()">播放/暂停</button>
  <button onclick="makeBig()">大</button>
  <button onclick="makeNormal()">中</button>
  <button onclick="makeSmall()">小</button>
  <br />
  <video id="video1" width="420" style="margin-top:15px;">
    <source src="/example/html5/mov_bbb.mp4" type="video/mp4" />
    <source src="/example/html5/mov_bbb.ogg" type="video/ogg" />
    Your browser does not support HTML5 video.
  </video>
</div>

<script type="text/javascript">
var myVideo=document.getElementById("video1");

function playPause()
{
if (myVideo.paused)
  myVideo.play();
else
  myVideo.pause();
}

function makeBig()
{
myVideo.width=560;
}

function makeSmall()
{
myVideo.width=320;
}

function makeNormal()
{
myVideo.width=420;
}
</script>

</body>
</html>
```

上面的例子调用了两个方法：play() 和 pause()。它同时使用了两个属性：paused 和 width。

HTML5 <video> - 方法、属性以及事件

下面列出了大多数浏览器支持的视频方法、属性和事件：

方法	属性	事件
play()	currentSrc	play
pause()	currentTime	pause
load()	videoWidth	progress
canPlayType	videoHeight	error
duration	timeupdate	
ended	ended	
error	abort	
paused	empty	
muted	emptied	
seeking	waiting	
volume	loadedmetadata	
height		
width		

注释：在所有属性中，只有 videoWidth 和 videoHeight 属性是立即可用的。在视频的元数据已加载后，其他属性才可用。

HTML 5 音频

HTML5 提供了播放音频的标准。

Web 上的音频

直到现在，仍然不存在一项旨在网页上播放音频的标准。

今天，大多数音频是通过插件（比如 Flash）来播放的。然而，并非所有浏览器都拥有同样的插件。

HTML5 规定了一种通过 audio 元素来包含音频的标准方法。

audio 元素能够播放声音文件或者音频流。

音频格式

当前，audio 元素支持三种音频格式：

IE 9	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis	√	√	√	
MP3	√	√	√	
Wav	√	√	√	

如何工作

如需在 HTML5 中播放音频，您所有需要的是：

```
<audio src="song.ogg" controls="controls">
</audio>
```

control 属性供添加播放、暂停和音量控件。

<audio> 与 </audio> 之间插入的内容是供不支持 audio 元素的浏览器显示的：

实例

```
<audio src="song.ogg" controls="controls">
Your browser does not support the audio tag.
</audio>
```

上面的例子使用一个 Ogg 文件，适用于Firefox、Opera 以及 Chrome 浏览器。

要确保适用于 Safari 浏览器，音频文件必须是 MP3 或 Wav 类型。

audio 元素允许多个 source 元素。source 元素可以链接不同的音频文件。浏览器将使用第一个可识别的格式：

实例

```
<audio controls="controls">
  <source src="song.ogg" type="audio/ogg">
  <source src="song.mp3" type="audio/mpeg">
Your browser does not support the audio tag.
</audio>
```

Internet Explorer

Internet Explorer 8 不支持 audio 元素。在 IE 9 中，将提供对 audio 元素的支持。

<audio> 标签的属性

属性	值	描述
autoplay	autoplay	如果出现该属性，则音频在就绪后马上播放。
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮。
loop	loop	如果出现该属性，则每当音频结束时重新开始播放。
preload	preload	如果出现该属性，则音频在页面加载时进行加载，并预备播放。 如果使用 "autoplay"，则忽略该属性。
src	*url*	要播放的音频的 URL。

相关页面

参考手册：HTML 5 <audio> 标签

HTML 5 拖放

拖放（**Drag** 和 **drop**）是 **HTML5** 标准的组成部分。

拖放

拖放是一种常见的特性，即抓取对象以后拖到另一个位置。

在 HTML5 中，拖放是标准的一部分，任何元素都能够拖放。

浏览器支持

Internet Explorer 9、Firefox、Opera 12、Chrome 以及 Safari 5 支持拖放。

注释：在 Safari 5.1.2 中不支持拖放。

HTML5 拖放实例

下面的例子是一个简单的拖放实例：

实例

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function allowDrop(ev)
{
ev.preventDefault();
}

function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}

function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69" />

</body>
</html>
```

它看上去也许有些复杂，不过我们可以分别研究拖放事件的不同部分。

设置元素为可拖放

首先，为了使元素可拖动，把 draggable 属性设置为 true ：

```
<img draggable="true" />
```

拖动什么 - ondragstart 和 setData()

然后，规定当元素被拖动时，会发生什么。

在上面的例子中，ondragstart 属性调用了一个函数，drag(*event*)，它规定了被拖动的数据。

dataTransfer.setData() 方法设置被拖数据的数据类型和值：

```
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

在这个例子中，数据类型是 "Text"，值是可拖动元素的 id ("drag1")。

放到何处 - ondragover

ondragover 事件规定在何处放置被拖动的数据。

默认地，无法将数据/元素放置到其他元素中。如果需要设置允许放置，我们必须阻止对元素的默认处理方式。

这要通过调用 ondragover 事件的 *event*.preventDefault() 方法：

```
event.preventDefault()
```

进行放置 - ondrop

当放置被拖数据时，会发生 drop 事件。

在上面的例子中，ondrop 属性调用了一个函数，drop(*event*)：

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
```

代码解释：

- 调用 preventDefault() 来避免浏览器对数据的默认处理（drop 事件的默认行为是以链接形式打开）
- 通过 dataTransfer.getData("Text") 方法获得被拖的数据。该方法将返回在 setData() 方法中设置为相同类型的任何数据。
- 被拖数据是被拖元素的 id ("drag1")
- 把被拖元素追加到放置元素（目标元素）中

更多实例

来回拖放图片

HTML 5 Canvas

canvas 元素用于在网页上绘制图形。

什么是 Canvas？

HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像。

画布是一个矩形区域，您可以控制其每一像素。

canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

创建 Canvas 元素

向 HTML5 页面添加 canvas 元素。

规定元素的 id、宽度和高度：

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

通过 JavaScript 来绘制

canvas 元素本身是没有绘图能力的。所有的绘制工作必须在 JavaScript 内部完成：

```
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
</script>
```

JavaScript 使用 id 来寻找 canvas 元素：

```
var c=document.getElementById("myCanvas");
```

然后，创建 context 对象：

```
var cxt=c.getContext("2d");
```

getContext("2d") 对象是内建的 HTML5 对象，拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

下面的两行代码绘制一个红色的矩形：

```
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
```

fillStyle 方法将其染成红色，fillRect 方法规定了形状、位置和尺寸。

理解坐标

上面的 fillRect 方法拥有参数 (0,0,150,75)。

意思是：在画布上绘制 150x75 的矩形，从左上角开始 (0,0)。

如下图所示，画布的 X 和 Y 坐标用于在画布上对绘画进行定位。



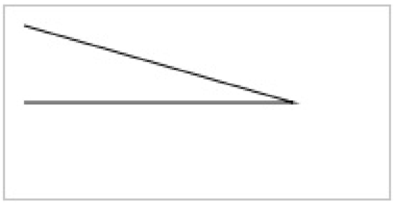
实例：把鼠标悬停在矩形上可以看到坐标

更多 Canvas 实例

下面的在 canvas 元素上进行绘画的更多实例：

实例 - 线条

通过指定从何处开始，在何处结束，来绘制一条线：



JavaScript 代码：

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.moveTo(10,10);
cxt.lineTo(150,50);
cxt.lineTo(10,50);
cxt.stroke();

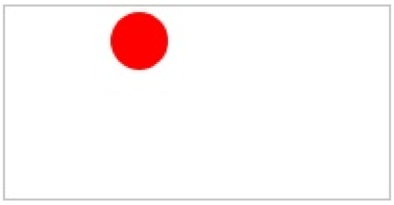
</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px sc
Your browser does not support the canvas element.
</canvas>
```

实例 - 圆形

通过规定尺寸、颜色和位置，来绘制一个圆：



JavaScript 代码：

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="#FF0000";
cxt.beginPath();
cxt.arc(70,18,15,0,Math.PI*2,true);
cxt.closePath();
cxt.fill();

</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px so
Your browser does not support the canvas element.
</canvas>
```

实例 - 渐变

使用您指定的颜色来绘制渐变背景：



JavaScript 代码：

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
var grd=cxt.createLinearGradient(0,0,175,50);
grd.addColorStop(0,"#FF0000");
grd.addColorStop(1,"#00FF00");
cxt.fillStyle=grd;
cxt.fillRect(0,0,175,50);

</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px so
Your browser does not support the canvas element.
</canvas>
```

实例 - 图像

把一幅图像放置到画布上：



JavaScript 代码：

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
var img=new Image()
img.src="flower.png"
cxt.drawImage(img,0,0);

</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px sc
Your browser does not support the canvas element.
</canvas>
```

相关页面

参考手册：HTML 5 <canvas> 标签

参考手册：HTML DOM Canvas 对象

HTML5 内联 SVG

HTML5 支持内联 SVG。

什么是SVG？

- SVG 指可伸缩矢量图形 (Scalable Vector Graphics)
- SVG 用于定义用于网络的基于矢量的图形
- SVG 使用 XML 格式定义图形
- SVG 图像在放大或改变尺寸的情况下其图形质量不会有损失
- SVG 是万维网联盟的标准

SVG 的优势

与其他图像格式相比（比如 JPEG 和 GIF），使用 SVG 的优势在于：

- SVG 图像可通过文本编辑器来创建和修改
- SVG 图像可被搜索、索引、脚本化或压缩
- SVG 是可伸缩的
- SVG 图像可在任何的分辨率下被高质量地打印
- SVG 可在图像质量不下降的情况下被放大

浏览器支持

Internet Explorer 9、Firefox、Opera、Chrome 以及 Safari 支持内联 SVG。

把 SVG 直接嵌入 HTML 页面

在 HTML5 中，您能够将 SVG 元素直接嵌入 HTML 页面中：

实例

```
<!DOCTYPE html>
<html>
<body>
 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190'
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;'
</svg>
</body>
</html>
```

结果：

如需学习更多有关 SVG 的知识，请阅读我们的 SVG 教程。

HTML 5 Canvas vs. SVG

Canvas 和 **SVG** 都允许您在浏览器中创建图形，但是它们在根本上是不同的。

SVG

SVG 是一种使用 XML 描述 2D 图形的语言。

SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。

在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。

Canvas

Canvas 通过 JavaScript 来绘制 2D 图形。

Canvas 是逐像素进行渲染的。

在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

Canvas 与 SVG 的比较

下表列出了 canvas 与 SVG 之间的一些不同之处。

Canvas

- 依赖分辨率
- 不支持事件处理器
- 弱的文本渲染能力
- 能够以 .png 或 .jpg 格式保存结果图像
- 最适合图像密集型的游戏，其中的许多对象会被频繁重绘

SVG

- 不依赖分辨率
- 支持事件处理器
- 最适合带有大型渲染区域的应用程序（比如谷歌地图）
- 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）
- 不适合游戏应用

HTML5 地理定位

HTML5 Geolocation（地理定位）用于定位用户的位置。

亲自试一试：在谷歌地图上显示您的位置

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">点击这个按钮，获得您的位置：</p>
<button onclick="getLocation()">试一下</button>
<div id="mapholder"></div>
<script src="http://maps.google.com/maps/api/js?sensor=false"></sc
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition,showErro
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";
  }

function showPosition(position)
  {
  lat=position.coords.latitude;
  lon=position.coords.longitude;
  latlon=new google.maps.LatLng(lat, lon)
  mapholder=document.getElementById('mapholder')
  mapholder.style.height='250px';
  mapholder.style.width='500px';

  var myOptions={
  center:latlon,zoom:14,
  mapTypeId:google.maps.MapTypeId.ROADMAP,
  mapTypeControl:false,
  navigationControlOptions:{style:google.maps.NavigationControlSty
  };
  var map=new google.maps.Map(document.getElementById("mapholder"),
  var marker=new google.maps.Marker({position:latlon,map:map,title
  }

function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="User denied the request for Geolocation."
```

```
        break;
      case error.POSITION_UNAVAILABLE:
        x.innerHTML="Location information is unavailable."
        break;
      case error.TIMEOUT:
        x.innerHTML="The request to get user location timed out."
        break;
      case error.UNKNOWN_ERROR:
        x.innerHTML="An unknown error occurred."
        break;
      }
    }
</script>
</body>
</html>
```

定位用户的位置

HTML5 Geolocation API 用于获得用户的地理位置。

鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

浏览器支持

Internet Explorer 9、Firefox、Chrome、Safari 以及 Opera 支持地理定位。

注释：对于拥有 GPS 的设备，比如 iPhone，地理定位更加精确。

HTML5 - 使用地理定位

请使用 getCurrentPosition() 方法来获得用户的位置。

下例是一个简单的地理定位实例，可返回用户位置的经度和纬度。

实例

```
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.coords.latitude +
  "<br />Longitude: " + position.coords.longitude;
  }
</script>
```

例子解释：

- 检测是否支持地理定位
- 如果支持，则运行 getCurrentPosition() 方法。如果不支持，则向用户显示一段消息。
- 如果getCurrentPosition()运行成功，则向参数showPosition中规定的函数返回一个coordinates对象
- showPosition() 函数获得并显示经度和纬度

上面的例子是一个非常基础的地理定位脚本，不含错误处理。

处理错误和拒绝

getCurrentPosition() 方法的第二个参数用于处理错误。它规定当获取用户位置失败时运行的函数：

实例

```
function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML="Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML="The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML="An unknown error occurred."
      break;
    }
  }
```

错误代码：

- Permission denied - 用户不允许地理定位
- Position unavailable - 无法获取当前位置
- Timeout - 操作超时

在地图中显示结果

如需在地图中显示结果，您需要访问可使用经纬度的地图服务，比如谷歌地图或百度地图：

实例

```
function showPosition(position)
{
var latlon=position.coords.latitude+","+position.coords.longitude;

var img_url="http://maps.googleapis.com/maps/api/staticmap?center='
+latlon+"&zoom=14&size=400x300&sensor=false";

document.getElementById("mapholder").innerHTML="<img src='"+img_ur
}
```

在上例中，我们使用返回的经纬度数据在谷歌地图中显示位置（使用静态图像）。

谷歌地图脚本

上面的链接向您演示如何使用脚本来显示带有标记、缩放和拖曳选项的交互式地图。

给定位置的信息

本页演示的是如何在地图上显示用户的位置。不过，地理定位对于给定位置的信息同样很有用处。

案例：

- 更新本地信息
- 显示用户周围的兴趣点
- 交互式车载导航系统 (GPS)

getCurrentPosition() 方法 - 返回数据

若成功，则 getCurrentPosition() 方法返回对象。始终会返回 latitude、longitude 以及 accuracy 属性。如果可用，则会返回其他下面的属性。

属性	描述
coords.latitude	十进制数的纬度
coords.longitude	十进制数的经度
coords.accuracy	位置精度
coords.altitude	海拔，海平面以上以米计
coords.altitudeAccuracy	位置的海拔精度
coords.heading	方向，从正北开始以度计
coords.speed	速度，以米/每秒计
timestamp	响应的日期/时间

Geolocation 对象 - 其他有趣的方法

watchPosition() - 返回用户的当前位置，并继续返回用户移动时的更新位置（就像汽车上的 GPS）。

clearWatch() - 停止 watchPosition() 方法

下面的例子展示 watchPosition() 方法。您需要一台精确的 GPS 设备来测试该例（比如 iPhone）：

实例

```
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.watchPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.coords.latitude +
  "<br />Longitude: " + position.coords.longitude;
  }
</script>
```

HTML 5 Web 存储

在客户端存储数据

HTML5 提供了两种在客户端存储数据的新方法：

- localStorage - 没有时间限制的数据存储
- sessionStorage - 针对一个 session 的数据存储

之前，这些都是由 cookie 完成的。但是 cookie 不适合大量数据的存储，因为它们由每个对服务器的请求来传递，这使得 cookie 速度很慢而且效率也不高。

在 HTML5 中，数据不是由每个服务器请求传递的，而是只有在请求时使用数据。它使在不影响网站性能的情况下存储大量数据成为可能。

对于不同的网站，数据存储于不同的区域，并且一个网站只能访问其自身的数据。

HTML5 使用 JavaScript 来存储和访问数据。

localStorage 方法

localStorage 方法存储的数据没有时间限制。第二天、第二周或下一年之后，数据依然可用。

如何创建和访问 localStorage：

实例

```
<script type="text/javascript">
localStorage.lastname="Smith";
document.write(localStorage.lastname);
</script>
```

下面的例子对用户访问页面的次数进行计数：

实例

```
<script type="text/javascript">
if (localStorage.pagecount)
  {
  localStorage.pagecount=Number(localStorage.pagecount) +1;
  }
else
  {
  localStorage.pagecount=1;
  }
document.write("Visits "+ localStorage.pagecount + " time(s).");
</script>
```

sessionStorage 方法

sessionStorage 方法针对一个 session 进行数据存储。当用户关闭浏览器窗口后，数据会被删除。

如何创建并访问一个 sessionStorage：

实例

```
<script type="text/javascript">
sessionStorage.lastname="Smith";
document.write(sessionStorage.lastname);
</script>
```

下面的例子对用户在当前 session 中访问页面的次数进行计数：

实例

```
<script type="text/javascript">
if (sessionStorage.pagecount)
  {
  sessionStorage.pagecount=Number(sessionStorage.pagecount) +1;
  }
else
  {
  sessionStorage.pagecount=1;
  }
document.write("Visits "+sessionStorage.pagecount+" time(s) this se
</script>
```

HTML 5 应用程序缓存

使用 **HTML5**，通过创建 **cache manifest** 文件，可以轻松地创建 **web** 应用的离线版本。

什么是应用程序缓存（**Application Cache**）？

HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。

应用程序缓存为应用带来三个优势：

- 离线浏览 - 用户可在应用离线时使用它们
- 速度 - 已缓存资源加载得更快
- 减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

浏览器支持

所有主流浏览器均支持应用程序缓存，除了 Internet Explorer。

HTML5 Cache Manifest 实例

下面的例子展示了带有 cache manifest 的 HTML 文档（供离线浏览）：

实例

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">

<body>
The content of the document......
</body>

</html>
```

Cache Manifest 基础

如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。

manifest 文件的建议的文件扩展名是：".appcache"。

请注意，manifest 文件需要配置正确的 *MIME-type*，即 "text/cache-manifest"。必须在 web 服务器上进行配置。

Manifest 文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件可分为三个部分：

- *CACHE MANIFEST* - 在此标题下列出的文件将在首次下载后进行缓存
- *NETWORK* - 在此标题下列出的文件需要与服务器的连接，且不会被缓存
- *FALLBACK* - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

CACHE MANIFEST

第一行，CACHE MANIFEST，是必需的：

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的。

NETWORK

下面的 NETWORK 小节规定文件 "login.asp" 永远不会被缓存，且离线时是不可用的：

```
NETWORK:
login.asp
```

可以使用星号来指示所有其他资源/文件都需要因特网连接：

```
NETWORK:
*
```

FALLBACK

下面的 FALLBACK 小节规定如果无法建立因特网连接，则用 "offline.html" 替代 /html5/ 目录中的所有文件：

```
FALLBACK:
/html5/ /404.html
```

注释：第一个 URI 是资源，第二个是替补。

更新缓存

一旦应用被缓存，它就会保持缓存直到发生下列情况：

- 用户清空浏览器缓存
- manifest 文件被修改（参阅下面的提示）
- 由程序来更新应用缓存

实例 - 完整的 Manifest 文件

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html5/ /404.html
```

重要的提示：以 "#" 开头的是注释行，但也可满足其他用途。应用的缓存会在其 manifest 文件更改时被更新。如果您编辑了一幅图片，或者修改了一个 JavaScript 函数，这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器

重新缓存文件的办法。

关于应用程序缓存的注释

请留心缓存的内容。

一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使您修改了服务器上的文件。为了确保浏览器更新缓存，您需要更新 manifest 文件。

注释：浏览器对缓存数据的容量限制可能不太一样（某些浏览器设置的限制是每个站点 5MB）。

HTML 5 Web Workers

web worker 是运行在后台的 **JavaScript**，不会影响页面的性能。

什么是 Web Worker？

当在 HTML 页面中执行脚本时，页面的状态是不可响应的，直到脚本已完成。

web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行。

浏览器支持

所有主流浏览器均支持 web worker，除了 Internet Explorer。

HTML5 Web Workers 实例

下面的例子创建了一个简单的 web worker，在后台计数：

计数：

<button onclick="startWorker()" style="font:12px Verdana, Arial, Helvetica, sans-serif;">启动 Worker</button> <button onclick="stopWorker()" style="font:12px Verdana, Arial, Helvetica, sans-serif;">停止 Worker</button>

检测 Web Worker 支持

在创建 web worker 之前，请检测用户的浏览器是否支持它：

```
if(typeof(Worker)!=="undefined")
  {
  // Yes! Web worker support!
  // Some code.....
  }
else
  {
  // Sorry! No Web Worker support..
  }
```

创建 web worker 文件

现在，让我们在一个外部 JavaScript 中创建我们的 web worker。

在这里，我们创建了计数脚本。该脚本存储于 "demo_workers.js" 文件中：

```
var i=0;

function timedCount()
{
i=i+1;
postMessage(i);
setTimeout("timedCount()",500);
}

timedCount();
```

以上代码中重要的部分是 *postMessage()* 方法 - 它用于向 HTML 页面传回一段消息。

注释：web worker 通常不用于如此简单的脚本，而是用于更耗费 CPU 资源的任务。

创建 Web Worker 对象

我们已经有了 web worker 文件，现在我们需要从 HTML 页面调用它。

下面的代码检测是否存在 worker，如果不存在，- 它会创建一个新的 web worker 对象，然后运行 "demo_workers.js" 中的代码：

```
if(typeof(w)=="undefined")
  {
  w=new Worker("demo_workers.js");
  }
```

然后我们就可以从 web worker 发生和接收消息了。

向 web worker 添加一个 "onmessage" 事件监听器：

```
w.onmessage=function(event){
document.getElementById("result").innerHTML=event.data;
};
```

当 web worker 传递消息时，会执行事件监听器中的代码。event.data 中存有来自 event.data 的数据。

终止 Web Worker

当我们创建 web worker 对象后，它会继续监听消息（即使在外部脚本完成之后）直到其被终止为止。

如需终止 web worker，并释放浏览器/计算机资源，请使用 terminate() 方法：

```
w.terminate();
```

完整的 Web Worker 实例代码

我们已经看到了 .js 文件中的 Worker 代码。下面是 HTML 页面的代码：

实例

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<br /><br />

<script>
var w;

function startWorker()
{
if(typeof(Worker)!=="undefined")
{
  if(typeof(w)=="undefined")
    {
    w=new Worker("demo_workers.js");
    }
  w.onmessage = function (event) {
    document.getElementById("result").innerHTML=event.data;
  };
}
else
{
document.getElementById("result").innerHTML="Sorry, your browser
 does not support Web Workers...";
}
}

function stopWorker()
{
w.terminate();
}
</script>

</body>
</html>
```

Web Workers 和 DOM

由于 web worker 位于外部文件中，它们无法访问下例 JavaScript 对象：

- window 对象
- document 对象
- parent 对象

HTML 5 服务器发送事件

HTML5 服务器发送事件（**server-sent event**）允许网页获得来自服务器的更新。

Server-Sent 事件 - 单向消息传递

Server-Sent 事件指的是网页自动获取来自服务器的更新。

以前也可能做到这一点，前提是网页不得不询问是否有可用的更新。通过服务器发送事件，更新能够自动到达。

例子：Facebook/Twitter 更新、估价更新、新的博文、赛事结果等。

浏览器支持

所有主流浏览器均支持服务器发送事件，除了 Internet Explorer。

接收 Server-Sent 事件通知

EventSource 对象用于接收服务器发送事件通知：

实例

```
var source=new EventSource("demo_sse.php");
source.onmessage=function(event)
  {
  document.getElementById("result").innerHTML+=event.data + "<br />
  };
```

例子解释：

- 创建一个新的 EventSource 对象，然后规定发送更新的页面的 URL（本例中是 "demo_sse.php"）
- 每接收到一次更新，就会发生 onmessage 事件
- 当 onmessage 事件发生时，把已接收的数据推入 id 为 "result" 的元素中

检测 Server-Sent 事件支持

在上面的 TIY 实例中，我们编写了一段额外的代码来检测服务器发送事件的浏览器支持情况：

```
if(typeof(EventSource)!=="undefined")
  {
  // Yes! Server-sent events support!
  // Some code.....
  }
else
  {
  // Sorry! No server-sent events support..
  }
```

服务器端代码实例

为了让上面的例子可以运行，您还需要能够发送数据更新的服务器（比如 PHP 和 ASP）。

服务器端事件流的语法是非常简单的。把 "Content-Type" 报头设置为 "text/event-stream"。现在，您可以开始发送事件流了。

PHP 代码 (demo_sse.php)：

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

ASP 代码 (VB) (demo_sse.asp):

```
<%
Response.ContentType="text/event-stream"
Response.Expires=-1
Response.Write("data: " & now())
Response.Flush()
%>
```

代码解释：

- 把报头 "Content-Type" 设置为 "text/event-stream"
- 规定不对页面进行缓存
- 输出发送日期（始终以 "data: " 开头）
- 向网页刷新输出数据

EventSource 对象

在上面的例子中，我们使用 onmessage 事件来获取消息。不过还可以使用其他事件：

事件	描述
onopen	当通往服务器的连接被打开
onmessage	当接收到消息
onerror	当错误发生

HTML5 Input 类型

HTML5 新的 Input 类型

HTML5 拥有多个新的表单输入类型。这些新特性提供了更好的输入控制和验证。

本章全面介绍这些新的输入类型：

- email
- url
- number
- range
- Date pickers (date, month, week, time, datetime, datetime-local)
- search
- color

浏览器支持

Input type	IE	Firefox	Opera	Chrome	Safari
email	No	4.0	9.0	10.0	No
url	No	4.0	9.0	10.0	No
number	No	No	9.0	7.0	No
range	No	No	9.0	4.0	4.0
Date pickers	No	No	9.0	10.0	No
search	No	4.0	11.0	10.0	No
color	No	No	11.0	No	No

注释：Opera 对新的输入类型的支持最好。不过您已经可以在所有主流的浏览器中使用它们了。即使不被支持，仍然可以显示为常规的文本域。

Input 类型 - email

email 类型用于应该包含 e-mail 地址的输入域。

在提交表单时，会自动验证 email 域的值。

实例

```
E-mail: <input type="email" name="user_email" />
```

提示：iPhone 中的 Safari 浏览器支持 email 输入类型，并通过改变触摸屏键盘来配合它（添加 @ 和 .com 选项）。

Input 类型 - url

url 类型用于应该包含 URL 地址的输入域。

在提交表单时，会自动验证 url 域的值。

实例

```
Homepage: <input type="url" name="user_url" />
```

提示：iPhone 中的 Safari 浏览器支持 url 输入类型，并通过改变触摸屏键盘来配合它（添加 .com 选项）。

Input 类型 - number

number 类型用于应该包含数值的输入域。

您还能够设定对所接受的数字的限定：

实例

```
Points: <input type="number" name="points" min="1" max="10" />
```

请使用下面的属性来规定对数字类型的限定：

属性	值	描述
max	*number*	规定允许的最大值
min	*number*	规定允许的最小值
step	*number*	规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value	*number*	规定默认值

请试一下带有所有限定属性的例子：

提示：iPhone 中的 Safari 浏览器支持 number 输入类型，并通过改变触摸屏键盘来配合它（显示数字）。

Input 类型 - range

range 类型用于应该包含一定范围内数字值的输入域。

range 类型显示为滑动条。

您还能够设定对所接受的数字的限定：

实例

```
<input type="range" name="points" min="1" max="10" />
```

请使用下面的属性来规定对数字类型的限定：

属性	值	描述
max	*number*	规定允许的最大值
min	*number*	规定允许的最小值
step	*number*	规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value	*number*	规定默认值

Input 类型 - Date Pickers（日期选择器）

HTML5 拥有多个可供选取日期和时间的新输入类型：

- date - 选取日、月、年
- month - 选取月、年
- week - 选取周和年
- time - 选取时间（小时和分钟）
- datetime - 选取时间、日、月、年（UTC 时间）
- datetime-local - 选取时间、日、月、年（本地时间）

下面的例子允许您从日历中选取一个日期：

实例

```
Date: <input type="date" name="user_date" />
```

输入类型 "month":

输入类型 "week":

输入类型 "time":

输入类型 "datetime":

输入类型 "datetime-local":

Input 类型 - search

search 类型用于搜索域，比如站点搜索或 Google 搜索。

search 域显示为常规的文本域。

HTML5 表单元素

HTML5 的新的表单元素：

HTML5 拥有若干涉及表单的元素和属性。

本章介绍以下新的表单元素：

- datalist
- keygen
- output

浏览器支持

Input type	IE	Firefox	Opera	Chrome	Safari
datalist	No	No	9.5	No	No
keygen	No	No	10.5	3.0	No
output	No	No	9.5	No	No

datalist 元素

datalist 元素规定输入域的选项列表。

列表是通过 datalist 内的 option 元素创建的。

如需把 datalist 绑定到输入域，请用输入域的 list 属性引用 datalist 的 id：

实例

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3School" value="http://www.W3School.com.cn" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

提示：option 元素永远都要设置 value 属性。

keygen 元素

keygen 元素的作用是提供一种验证用户的可靠方法。

keygen 元素是密钥对生成器（key-pair generator）。当提交表单时，会生成两个键，一个是私钥，一个公钥。

私钥（private key）存储于客户端，公钥（public key）则被发送到服务器。公钥可用于之后验证用户的客户端证书（client certificate）。

目前，浏览器对此元素的糟糕的支持度不足以使其成为一种有用的安全标准。

实例

```
<form action="demo_form.asp" method="get">
Username: <input type="text" name="usr_name" />
Encryption: <keygen name="security" />
<input type="submit" />
</form>
```

output 元素

output 元素用于不同类型的输出，比如计算或脚本输出：

实例

```
<output id="result" onforminput="resCalc()"></output>
```

HTML5 表单属性

HTML5 的新的表单属性

本章讲解涉及 <form> 和 <input> 元素的新属性。

新的 **form** 属性：

- autocomplete
- novalidate

新的 **input** 属性：

- autocomplete
- autofocus
- form
- form overrides (formaction, formenctype, formmethod, formnovalidate, formtarget)
- height 和 width
- list
- min, max 和 step
- multiple
- pattern (regexp)
- placeholder
- required

浏览器支持

Input type	IE	Firefox	Opera	Chrome	Safari
autocomplete	8.0	3.5	9.5	3.0	4.0
autofocus	No	No	10.0	3.0	4.0
form	No	No	9.5	No	No
form overrides	No	No	10.5	No	No
height and width	8.0	3.5	9.5	3.0	4.0
list	No	No	9.5	No	No
min, max and step	No	No	9.5	3.0	No
multiple	No	3.5	No	3.0	4.0
novalidate	No	No	No	No	No
pattern	No	No	9.5	3.0	No
placeholder	No	No	No	3.0	3.0
required	No	No	9.5	3.0	No

autocomplete 属性

autocomplete 属性规定 form 或 input 域应该拥有自动完成功能。

注释：autocomplete 适用于 <form> 标签，以及以下类型的 <input> 标签：text, search, url, telephone, email, password, datepickers, range 以及 color。

当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项：

实例

```
<form action="demo_form.asp" method="get" autocomplete="on">
First name: <input type="text" name="fname" /><br />
Last name: <input type="text" name="lname" /><br />
E-mail: <input type="email" name="email" autocomplete="off" /><br /
<input type="submit" />
</form>
```

注释：在某些浏览器中，您可能需要启用自动完成功能，以使该属性生效。

autofocus 属性

autofocus 属性规定在页面加载时，域自动地获得焦点。

注释：autofocus 属性适用于所有 <input> 标签的类型。

实例

```
User name: <input type="text" name="user_name"  autofocus="autofocu
```

form 属性

form 属性规定输入域所属的一个或多个表单。

注释：form 属性适用于所有 <input> 标签的类型。

form 属性必须引用所属表单的 id：

实例

```
<form action="demo_form.asp" method="get" id="user_form">
First name:<input type="text" name="fname" />
<input type="submit" />
</form>
Last name: <input type="text" name="lname" form="user_form" />
```

注释：如需引用一个以上的表单，请使用空格分隔的列表。

表单重写属性

表单重写属性（form override attributes）允许您重写 form 元素的某些属性设定。

表单重写属性有：

- formaction - 重写表单的 action 属性
- formenctype - 重写表单的 enctype 属性
- formmethod - 重写表单的 method 属性
- formnovalidate - 重写表单的 novalidate 属性
- formtarget - 重写表单的 target 属性

注释：表单重写属性适用于以下类型的 <input> 标签：submit 和 image。

实例

```
<form action="demo_form.asp" method="get" id="user_form">
E-mail: <input type="email" name="userid" /><br />
<input type="submit" value="Submit" />
<br />
<input type="submit" formaction="demo_admin.asp" value="Submit as a
<br />
<input type="submit" formnovalidate="true" value="Submit without va
<br />
</form>
```

注释：这些属性对于创建不同的提交按钮很有帮助。

height 和 width 属性

height 和 width 属性规定用于 image 类型的 input 标签的图像高度和宽度。

注释：height 和 width 属性只适用于 image 类型的 <input> 标签。

实例

```
<input type="image" src="img_submit.gif" width="99" height="99" />
```

list 属性

list 属性规定输入域的 datalist。datalist 是输入域的选项列表。

注释：list 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, date pickers, number, range 以及 color。

实例

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3Schools" value="http://www.w3school.com.cn" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

min、max 和 step 属性

min、max 和 step 属性用于为包含数字或日期的 input 类型规定限定（约束）。

max 属性规定输入域所允许的最大值。

min 属性规定输入域所允许的最小值。

step 属性为输入域规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）。

注释：min、max 和 step 属性适用于以下类型的 <input> 标签：date pickers、number 以及 range。

下面的例子显示一个数字域，该域接受介于 0 到 10 之间的值，且步进为 3（即合法的值为 0、3、6 和 9）：

实例

```
Points: <input type="number" name="points" min="0" max="10" step="3
```

multiple 属性

multiple 属性规定输入域中可选择多个值。

注释：multiple 属性适用于以下类型的 <input> 标签：email 和 file。

实例

```
Select images: <input type="file" name="img" multiple="multiple" />
```

novalidate 属性

novalidate 属性规定在提交表单时不应该验证 form 或 input 域。

注释：novalidate 属性适用于 <form> 以及以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, range 以及 color.

实例

```
<form action="demo_form.asp" method="get" novalidate="true">
E-mail: <input type="email" name="user_email" />
<input type="submit" />
</form>
```

pattern 属性

pattern 属性规定用于验证 input 域的模式（pattern）。

模式（pattern） 是正则表达式。您可以在我们的 JavaScript 教程中学习到有关正则表达式的内容。

注释：pattern 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

下面的例子显示了一个只能包含三个字母的文本域（不含数字及特殊字符）：

实例

```
Country code: <input type="text" name="country_code"
pattern="[A-z]{3}" title="Three letter country code" />
```

placeholder 属性

placeholder 属性提供一种提示（hint），描述输入域所期待的值。

注释：placeholder 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

提示（hint）会在输入域为空时显示出现，会在输入域获得焦点时消失：

实例

```
<input type="search" name="user_search"  placeholder="Search W3Scho
```

required 属性

required 属性规定必须在提交之前填写输入域（不能为空）。

注释：required 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, number, checkbox, radio 以及 file。

实例

```
Name: <input type="text" name="usr_name" required="required" />
```

W3School HTML 参考手册

来源：HTML 参考手册

整理：飞龙

HTML 标签列表

按字母顺序排列

New : HTML5 中的新标签。

标签	描述
<!--...-->	定义注释。
<!DOCTYPE>	定义文档类型。
<a>	定义锚。
<abbr>	定义缩写。
<acronym>	定义只取首字母的缩写。
<address>	定义文档作者或拥有者的联系信息。
<applet>	不赞成使用。定义嵌入的 applet。
<area>	定义图像映射内部的区域。
<article>	定义文章。
<aside>	定义页面内容之外的内容。
<audio>	定义声音内容。
<b>	定义粗体字。
<base>	定义页面中所有链接的默认地址或默认目标。
<basefont>	不赞成使用。定义页面中文本的默认字体、颜色或尺寸。
<bdi>	定义文本的文本方向，使其脱离其周围文本的方向设置。
<bdo>	定义文字方向。
<big>	定义大号文本。
<blockquote>	定义长的引用。
<body>	定义文档的主体。
 	定义简单的折行。
<button>	定义按钮 (push button)。
<canvas>	定义图形。
<caption>	定义表格标题。

<center>	不赞成使用。定义居中文本。
<cite>	定义引用(citation)。
<code>	定义计算机代码文本。
<col>	定义表格中一个或多个列的属性值。
<colgroup>	定义表格中供格式化的列组。
<command>	定义命令按钮。
<datalist>	定义下拉列表。
<dd>	定义定义列表中项目的描述。
<del>	定义被删除文本。
<details>	定义元素的细节。
<dir>	不赞成使用。定义目录列表。
<div>	定义文档中的节。
<dfn>	定义定义项目。
<dialog>	定义对话框或窗口。
<dl>	定义定义列表。
<dt>	定义定义列表中的项目。
<em>	定义强调文本。
<embed>	定义外部交互内容或插件。
<fieldset>	定义围绕表单中元素的边框。
<figcaption>	定义 figure 元素的标题。
<figure>	定义媒介内容的分组，以及它们的标题。
<font>	不赞成使用。定义文字的字体、尺寸和颜色。
<footer>	定义 section 或 page 的页脚。
<form>	定义供用户输入的 HTML 表单。
<frame>	定义框架集的窗口或框架。
<frameset>	定义框架集。
<h1> to <h6>	定义 HTML 标题。
<head>	定义关于文档的信息。
<header>	定义 section 或 page 的页眉。
<hr>	定义水平线。

<html>	定义 HTML 文档。
<i>	定义斜体字。
<iframe>	定义内联框架。
<img>	定义图像。
<input>	定义输入控件。
<ins>	定义被插入文本。
<isindex>	不赞成使用。定义与文档相关的可搜索索引。
<kbd>	定义键盘文本。
<keygen>	定义生成密钥。
<label>	定义 input 元素的标注。
<legend>	定义 fieldset 元素的标题。
	定义列表的项目。
<link>	定义文档与外部资源的关系。
<map>	定义图像映射。
<mark>	定义有记号的文本。
<menu>	定义命令的列表或菜单。
<menuitem>	定义用户可以从弹出菜单调用的命令/菜单项目。
<meta>	定义关于 HTML 文档的元信息。
<meter>	定义预定义范围内的度量。
<nav>	定义导航链接。
<noframes>	定义针对不支持框架的用户的替代内容。
<noscript>	定义针对不支持客户端脚本的用户的替代内容。
<object>	定义内嵌对象。
	定义有序列表。
<optgroup>	定义选择列表中相关选项的组合。
<option>	定义选择列表中的选项。
<output>	定义输出的一些类型。
<p>	定义段落。
<param>	定义对象的参数。
<pre>	定义预格式文本。

<progress>	定义任何类型的任务的进度。
<q>	定义短的引用。
<rp>	定义若浏览器不支持 ruby 元素显示的内容。
<rt>	定义 ruby 注释的解释。
<ruby>	定义 ruby 注释。
<s>	不赞成使用。定义加删除线的文本。
<samp>	定义计算机代码样本。
<script>	定义客户端脚本。
<section>	定义 section。
<select>	定义选择列表（下拉列表）。
<small>	定义小号文本。
<source>	定义媒介源。
<span>	定义文档中的节。
<strike>	不赞成使用。定义加删除线文本。
<strong>	定义强调文本。
<style>	定义文档的样式信息。
<sub>	定义下标文本。
<summary>	为 <details> 元素定义可见的标题。
<sup>	定义上标文本。
<table>	定义表格。
<tbody>	定义表格中的主体内容。
<td>	定义表格中的单元。
<textarea>	定义多行的文本输入控件。
<tfoot>	定义表格中的表注内容（脚注）。
<th>	定义表格中的表头单元格。
<thead>	定义表格中的表头内容。
<time>	定义日期/时间。
<title>	定义文档的标题。
<tr>	定义表格中的行。

<track>	定义用在媒体播放器中的文本轨道。
<tt>	定义打字机文本。
<u>	不赞成使用。定义下划线文本。
	定义无序列表。
<var>	定义文本的变量部分。
<video>	定义视频。
<wbr>	定义视频。
<xmp>	不赞成使用。定义预格式文本。

按功能类别排列

New : HTML5 中的新标签。

基础

标签	描述
<!DOCTYPE>	定义文档类型。
<html>	定义 HTML 文档。
<title>	定义文档的标题。
<body>	定义文档的主体。
<h1> to <h6>	定义 HTML 标题。
<p>	定义段落。
 	定义简单的折行。
<hr>	定义水平线。
<!--...-->	定义注释。

格式

标签	描述
<acronym>	定义只取首字母的缩写。
<abbr>	定义缩写。
<address>	定义文档作者或拥有者的联系信息。

<b>	定义粗体文本。
<bdi>	定义文本的文本方向，使其脱离其周围文本的方向设置。
<bdo>	定义文字方向。
<big>	定义大号文本。
<blockquote>	定义长的引用。
<center>	不赞成使用。定义居中文本。
<cite>	定义引用(citation)。
<code>	定义计算机代码文本。
<del>	定义被删除文本。
<dfn>	定义定义项目。
<em>	定义强调文本。
<font>	不赞成使用。定义文本的字体、尺寸和颜色
<i>	定义斜体文本。
<ins>	定义被插入文本。
<kbd>	定义键盘文本。
<mark>	定义有记号的文本。
<meter>	定义预定义范围内的度量。
<pre>	定义预格式文本。
<progress>	定义任何类型的任务的进度。
<q>	定义短的引用。
<rp>	定义若浏览器不支持 ruby 元素显示的内容。
<rt>	定义 ruby 注释的解释。
<ruby>	定义 ruby 注释。
<s>	不赞成使用。定义加删除线的文本。
<samp>	定义计算机代码样本。
<small>	定义小号文本。
<strike>	不赞成使用。定义加删除线文本。
<strong>	定义语气更为强烈的强调文本。
<sup>	定义上标文本。
<sub>	定义下标文本。

<time>	定义日期/时间。
<tt>	定义打字机文本。
<u>	不赞成使用。定义下划线文本。
<var>	定义文本的变量部分。
<wbr>	定义视频。

表单

标签	描述
<form>	定义供用户输入的 HTML 表单。
<input>	定义输入控件。
<textarea>	定义多行的文本输入控件。
<button>	定义按钮。
<select>	定义选择列表（下拉列表）。
<optgroup>	定义选择列表中相关选项的组合。
<option>	定义选择列表中的选项。
<label>	定义 input 元素的标注。
<fieldset>	定义围绕表单中元素的边框。
<legend>	定义 fieldset 元素的标题。
<isindex>	不赞成使用。定义与文档相关的可搜索索引。
<datalist>	定义下拉列表。
<keygen>	定义生成密钥。
<output>	定义输出的一些类型。

框架

标签	描述
<frame>	定义框架集的窗口或框架。
<frameset>	定义框架集。
<noframes>	定义针对不支持框架的用户的替代内容。
<iframe>	定义内联框架。

图像

标签	描述
<img>	定义图像。
<map>	定义图像映射。
<area>	定义图像地图内部的区域。
<canvas>	定义图形。
<figcaption>	定义 figure 元素的标题。
<figure>	定义媒介内容的分组，以及它们的标题。

音频/视频

标签	描述
<audio>	定义声音内容。
<source>	定义媒介源。
<track>	定义用在媒体播放器中的文本轨道。
<video>	定义视频。

链接

标签	描述
<a>	定义锚。
<link>	定义文档与外部资源的关系。
<nav>	定义导航链接。

列表

标签	描述
	定义无序列表。
	定义有序列表。
	定义列表的项目。
<dir>	不赞成使用。定义目录列表。
<dl>	定义定义列表。
<dt>	定义定义列表中的项目。
<dd>	定义定义列表中项目的描述。
<menu>	定义命令的菜单/列表。
<menuitem>	定义用户可以从弹出菜单调用的命令/菜单项目。
<command>	定义命令按钮。

表格

标签	描述
<table>	定义表格
<caption>	定义表格标题。
<th>	定义表格中的表头单元格。
<tr>	定义表格中的行。
<td>	定义表格中的单元。
<thead>	定义表格中的表头内容。
<tbody>	定义表格中的主体内容。
<tfoot>	定义表格中的表注内容（脚注）。
<col>	定义表格中一个或多个列的属性值。
<colgroup>	定义表格中供格式化的列组。

样式/节

标签	描述
<style>	定义文档的样式信息。
<div>	定义文档中的节。
<span>	定义文档中的节。
<header>	定义 section 或 page 的页眉。
<footer>	定义 section 或 page 的页脚。
<section>	定义 section。
<article>	定义文章。
<aside>	定义页面内容之外的内容。
<details>	定义元素的细节。
<dialog>	定义对话框或窗口。
<summary>	为 <details> 元素定义可见的标题。

元信息

标签	描述
<head>	定义关于文档的信息。
<meta>	定义关于 HTML 文档的元信息。
<base>	定义页面中所有链接的默认地址或默认目标。
<basefont>	不赞成使用。定义页面中文本的默认字体、颜色或尺寸。

编程

标签	描述
<script>	定义客户端脚本。
<noscript>	定义针对不支持客户端脚本的用户的替代内容。
<applet>	不赞成使用。定义嵌入的 applet。
<embed>	为外部应用程序（非 HTML）定义容器。
<object>	定义嵌入的对象。
<param>	定义对象的参数。

HTML <!--...--> 标签

实例

HTML 注释：

```
<!--这是一段注释。注释不会在浏览器中显示。-->

<p>这是一段普通的段落。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持注释标签。

定义和用法

注释标签用于在源代码中插入注释。注释不会显示在浏览器中。

您可使用注释对您的代码进行解释，这样做有助于您在以后的时间对代码的编辑。当您编写了大量代码时尤其有用。

使用注释标签来隐藏浏览器不支持的脚本也是一个好习惯（这样就不会把脚本显示为纯文本）：

```
<script type="text/javascript">
<!--
function displayMsg()
{
alert("Hello World!")
}
//-->
</script>
```

注释：注释行结尾处的两条斜杠 (//) 是 JavaScript 注释符号。这可以避免 JavaScript 执行 --> 标签。

HTML 4.01 与 HTML5 之间的差异

无。

标准属性

注释标签不支持任何标准属性。

如需更多有关 HTML 标准属性的信息，请访问 标准属性。

事件属性

注释标签不支持任何事件属性。

如需更多有关 HTML 事件属性的信息，请访问 事件属性。

HTML <!DOCTYPE> 标签

实例

```
<!DOCTYPE html>
<html>
<head>
<title>文档的标题</title>
</head>

<body>
文档的内容......
</body>

</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <!DOCTYPE> 声明。

定义和用法

<!DOCTYPE> 声明必须是 HTML 文档的第一行，位于 <html> 标签之前。

<!DOCTYPE> 声明不是 HTML 标签；它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。

在 HTML 4.01 中，<!DOCTYPE> 声明引用 DTD，因为 HTML 4.01 基于 SGML。DTD 规定了标记语言的规则，这样浏览器才能正确地呈现内容。

HTML5 不基于 SGML，所以不需要引用 DTD。

提示：请始终向 HTML 文档添加 <!DOCTYPE> 声明，这样浏览器才能获知文档类型。

HTML 4.01 与 HTML5 之间的差异

在 HTML 4.01 中有三种 <!DOCTYPE> 声明。在 HTML5 中只有一种：

```
<!DOCTYPE html>
```

HTML 元素和文档类型（Doctype）

请参阅这个 HTML 元素表，其中列出了每种元素会出现在哪个文档类型中。

提示和注释

注释：<!DOCTYPE> 声明没有结束标签。

提示：<!DOCTYPE> 声明对大小写不敏感。

提示：请使用 W3C 的验证器来检查您是否编写了有效的 HTML / XHTML 文档！

常用的 DOCTYPE 声明

HTML 5

```
<!DOCTYPE html>
```

HTML 4.01 Strict

该 DTD 包含所有 HTML 元素和属性，但不包括展示性的和弃用的元素（比如 font）。不允许框架集（Framesets）。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.or
```

HTML 4.01 Transitional

该 DTD 包含所有 HTML 元素和属性，包括展示性的和弃用的元素（比如 font）。不允许框架集（Framesets）。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

HTML 4.01 Frameset

该 DTD 等同于 HTML 4.01 Transitional，但允许框架集内容。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML 1.0 Strict

该 DTD 包含所有 HTML 元素和属性，但不包括展示性的和弃用的元素（比如 font）。不允许框架集（Framesets）。必须以格式正确的 XML 来编写标记。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

该 DTD 包含所有 HTML 元素和属性，包括展示性的和弃用的元素（比如 font）。不允许框架集（Framesets）。必须以格式正确的 XML 来编写标记。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset

该 DTD 等同于 XHTML 1.0 Transitional，但允许框架集内容。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.1

该 DTD 等同于 XHTML 1.0 Strict，但允许添加模型（例如提供对东亚语系的 ruby 支持）。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.o
```

HTML <a> 标签

实例

指向 w3school 的超链接：

```
<a href="http://www.w3school.com.cn">W3School</a>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <a> 标签。

定义和用法

<a> 标签定义超链接，用于从一张页面链接到另一张页面。

<a> 元素最重要的属性是 href 属性，它指示链接的目标。

在所有浏览器中，链接的默认外观是：

- 未被访问的链接带有下划线而且是蓝色的
- 已被访问的链接带有下划线而且是紫色的
- 活动链接带有下划线而且是红色的

提示：您可能已经注意到了，W3School 站点内的链接外观与默认的链接外观非常不同。您可以使用 CSS 伪类 向文本超链接添加复杂而多样的样式。

术语解释

什么是超文本？

提示和注释

提示：如果不使用 href 属性，则不可以使用如下属性：download, hreflang, media, rel, target 以及 type 属性。

提示：被链接页面通常显示在当前浏览器窗口中，除非您规定了另一个目标（target 属性）。

提示：请使用 CSS 来设置链接的样式。

HTML 4.01 与 HTML5 之间的差异

在 HTML 4.01 中，<a> 标签可以是超链接或锚。在 HTML5 中，<a> 标签始终是超链接，但是如果未设置 href 属性，则只是超链接的占位符。

HTML5 提供了一些新属性，同时不再支持一些 HTML 4.01 属性。

属性

New：HTML5 中的新属性。

属性	值	描述
charset	*char_encoding*	HTML5 中不支持。规定被链接文档的字符集。
coords	*coordinates*	HTML5 中不支持。规定链接的坐标。
download	*filename*	规定被下载的超链接目标。
href	*URL*	规定链接指向的页面的 URL。
hreflang	*language_code*	规定被链接文档的语言。
media	*media_query*	规定被链接文档是为何种媒介/设备优化的。
name	*section_name*	HTML5 中不支持。规定锚的名称。
rel	*text*	规定当前文档与被链接文档之间的关系。
rev	*text*	HTML5 中不支持。规定被链接文档与当前文档之间的关系。
shape	`default` `rect` `circle` `poly`	HTML5 中不支持。规定链接的形状。
target	`_blank` `_parent` `_self` `_top` `framename`	规定在何处打开链接文档。
type	*MIME type*	规定被链接文档的的 MIME 类型。

全局属性

<a> 标签支持 HTML 中的全局属性。

事件属性

<a> 标签支持 HTML 中的事件属性。

亲自试一试 - 实例

创建超级链接

本例演示如何在 HTML 文档中创建链接。

```
<html>

<body>

<p>
<a href="/index.html">本文本</a> 是一个指向本网站中的一个页面的链接。</p>

<p><a href="http://www.microsoft.com/">本文本</a> 是一个指向万维网上的

</body>
</html>
```

将图像作为链接

本例演示如何使用图像作为链接。

```
<html>

<body>
<p>
您也可以使用图像来作链接：
<a href="/example/html/lastpage.html">
<img border="0" src="/i/eg_buttonnext.gif" />
</a>
</p>

</body>
</html>
```

链接到同一个页面的不同位置

本例演示如何使用链接跳转至文档的另一个部分

```
<html>

<body>

<p>
<a href="#C4">查看 Chapter 4。</a>
</p>

<h2>Chapter 1</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 2</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 3</h2>
<p>This chapter explains ba bla bla</p>

<h2><a name="C4">Chapter 4</a></h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 5</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 6</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 7</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 8</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 9</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 10</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 11</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 12</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 13</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 14</h2>
<p>This chapter explains ba bla bla</p>
```

```
<h2>Chapter 15</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 16</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 17</h2>
<p>This chapter explains ba bla bla</p>

</body>
</html>
```

在新的浏览器窗口打开链接

本例演示如何在新窗口打开一个页面，这样的话访问者就无需离开你的站点了。

```
<html>

<body>

<p>
<a href="#C4">查看 Chapter 4。</a>
</p>

<h2>Chapter 1</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 2</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 3</h2>
<p>This chapter explains ba bla bla</p>

<h2><a name="C4">Chapter 4</a></h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 5</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 6</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 7</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 8</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 9</h2>
<p>This chapter explains ba bla bla</p>
```

```
<h2>Chapter 10</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 11</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 12</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 13</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 14</h2><html>

<body>

<p>
<a href="#C4">查看 Chapter 4。</a>
</p>

<h2>Chapter 1</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 2</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 3</h2>
<p>This chapter explains ba bla bla</p>

<h2><a name="C4">Chapter 4</a></h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 5</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 6</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 7</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 8</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 9</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 10</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 11</h2>
<p>This chapter explains ba bla bla</p>
```

```
<h2>Chapter 12</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 13</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 14</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 15</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 16</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 17</h2>
<p>This chapter explains ba bla bla</p>

</body>
</html>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 15</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 16</h2>
<p>This chapter explains ba bla bla</p>

<h2>Chapter 17</h2>
<p>This chapter explains ba bla bla</p>

</body>
</html>
```

跳出框架

本例演示如何跳出框架，假如你的页面被固定在框架之内。

```
<html>

<body>

<p>被锁在框架中了吗？</p>

<a href="/index.html"
target="_top">请点击这里！</a>

</body>
</html>
```

创建电子邮件链接

本例演示如何如何链接到一个邮件。（本例在安装邮件客户端程序后才能工作。）

```
<html>

<body>

<p>
这是邮件链接：
<a href="mailto:someone@microsoft.com?subject=Hello%20again">发送邮件
</p>

<p>
<b>注意：</b>应该使用 %20 来替换单词之间的空格，这样浏览器就可以正确地显示文本
</p>

</body>
</html>
```

创建电子邮件链接 2

本例演示更加复杂的邮件链接。

```
<html>

<body>

<p>
这是另一个 mailto 链接：
<a href="mailto:someone@microsoft.com?cc=someoneelse@microsoft.com&
</p>

<p>
<b>注意：</b>应该使用 %20 来替换单词之间的空格，这样浏览器就可以正确地显示文本
</p>

</body>
</html>
```

相关页面

HTML 教程：HTML 链接

HTML DOM 参考手册：Anchor 对象

HTML <abbr> 标签

实例

标记一个缩写：

```
The <abbr title="People's Republic of China">PRC</abbr> was founded
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <abbr> 标签。

注释：IE 6 或更早版本的 IE 浏览器不支持 <abbr> 标签。

定义和用法

<abbr> 标签指示简称或缩写，比如 "WWW" 或 "NATO"。

通过对缩写进行标记，您能够为浏览器、拼写检查和搜索引擎提供有用的信息。

<abbr> 标签最初是在 HTML 4.0 中引入的，表示它所包含的文本是一个更长的单词或短语的缩写形式。

提示和注释

提示：可以在 <abbr> 标签中使用全局的 title 属性，这样就能够在鼠标指针移动到 <abbr> 元素上时显示出简称/缩写的完整版本。

HTML 4.01 与 HTML5 之间的差异

无。

全局属性

<abbr> 标签支持 HTML 中的全局属性。

事件属性

<abbr> 标签支持 HTML 中的事件属性。

HTML <acronym> 标签

实例

标记一个首字母缩写：

```
<acronym title="World Wide Web">WWW</acronym>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流的浏览器均支持 <acronym> 标签。

注释：IE 5.5 或更早版本的 IE 浏览器不支持 <acronym> 标签。

定义和用法

HTML5 中不支持 <acronym> 标签。请使用 <abbr> 标签代替。

<acronym> 标签定义首字母缩写。

缩写能够作为单词来朗读，例如 NATO, NASA, ASAP, GUI。

通过对缩写进行标记，您能够为浏览器、拼写检查和搜索引擎提供有用的信息。

提示和注释

提示：title 属性可用于在鼠标指针移动到元素上时显示出缩写的完整版本。

HTML 4.01 与 HTML5 之间的差异

HTML5 中不支持 <acronym> 标签，但是在 HTML 4.01 中支持。

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

TIY 实例

缩写和首字母缩写

此例演示如何实现缩写或首字母缩写。

```
<html>

<body>

<abbr title="etcetera">etc.</abbr>
<br />
<acronym title="World Wide Web">WWW</acronym>

<p>在某些浏览器中，当您把鼠标移至缩略词语上时，title 可用于展示表达的完整版本。

<p>仅对于 IE 5 中的 acronym 元素有效。</p>

<p>对于 Netscape 6.2 中的 abbr 和 acronym 元素都有效。</p>

</body>
</html>
```

HTML <address> 标签

实例

Example.com 的联系信息：

```
<address>
Written by <a href="mailto:webmaster@example.com">Donald Duck</a>.<
Visit us at:<br>
Example.com<br>
Box 564, Disneyland<br>
USA
</address>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <address> 标签。

定义和用法

<address> 标签定义文档或文章的作者/拥有者的联系信息。

如果 <address> 元素位于 <body> 元素内，则它表示文档联系信息。

如果 <address> 元素位于 <article> 元素内，则它表示文章的联系信息。

<address> 元素中的文本通常呈现为斜体。大多数浏览器会在 address 元素前后添加折行。

提示和注释

提示：<address> 标签不应该用于描述通讯地址，除非它是联系信息的一部分。

提示：<address> 元素通常连同其他信息被包含在 <footer> 元素中。

HTML 4.01 与 HTML5 之间的差异

HTML 4.01 不支持 <article> 标签，因此在 HTML 4.01 中，<address> 标签始终定义文档作者/拥有者的联系信息。

全局属性

<address> 标签支持 HTML 中的全局属性。

事件属性

<address> 标签支持 HTML 中的事件属性。

HTML <applet> 标签

实例

一个嵌入的 Java applet：

```
<applet code="Bubbles.class" width="350" height="350">
Java applet that draws animated bubbles.
</applet>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

某些浏览器中依然存在对 <applet> 标签的一些支持，但是需要额外的插件和安装过程才能起作用。

定义和用法

HTML5 中不支持 <applet> 标签。请使用 object 元素 标签代替。

HTML 4.01 中不赞成使用 <applet> 元素。

<applet> 标签定义嵌入的 applet。

HTML 4.01 与 HTML5 之间的差异

在 HTML5 中，不支持 applet 元素；在 HTML 4.01 中，不赞成使用 applet 元素。

必需的属性

属性	值	描述
code	*URL*	规定 Java applet 的文件名。
object	*name*	定义了包含该 applet 的一系列版本的资源名称。

可选的属性

属性	值	描述
align	`left` `right` `top` `bottom` `middle` `baseline` `texttop` `absmiddle` `absbottom`	定义 applet 相对于周围元素的对齐方式。
alt	*text*	规定 applet 的替换文本。
archive	*URL*	规定档案文件的位置。
codebase	*URL*	规定 code 属性中指定的 applet 的基准 URL。
height	*pixels*	定义 applet 的高度
hspace	*pixels*	定义围绕 applet 的水平间隔。
name	*unique_name*	规定 applet 的名称（用在脚本中的）。
vspace	*pixels*	定义围绕 applet 的垂直间隔。
width	*pixels*	定义 applet 的宽度

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

HTML <area> 标签

实例

带有可点击区域的图像映射：

```
<img src="planets.jpg" border="0" usemap="#planetmap" alt="Planets'

<map name="planetmap" id="planetmap">
  <area shape="circle" coords="180,139,14" href ="venus.html" alt='
  <area shape="circle" coords="129,161,10" href ="mercur.html" alt=
  <area shape="rect" coords="0,0,110,260" href ="sun.html" alt="Sur
</map>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <area> 标签。

定义和用法

<area> 标签定义图像映射中的区域（注：图像映射指得是带有可点击区域的图像）。

area 元素总是嵌套在 <map> 标签中。

注释：<img> 标签中的 usemap 属性与 map 元素 name 属性相关联，创建图像与映射之间的联系。

HTML 与 XHTML 之间的差异

在 HTML 中，<area> 没有结束标签。

在 XHTML 中，<area> 必须正确地关闭。

提示和注释：

注释：<img> 中的 *usemap* 属性可引用 <map> 中的 *id* 或 *name* 属性（由浏览器决定），所以我们需要同时向 <map> 添加 id 和 name 两个属性。

必需的属性

属性	值	描述
alt	*text*	定义此区域的替换文本。

可选的属性

属性	值	描述
coords	坐标值	定义可点击区域（对鼠标敏感的区域）的坐标。
href	*URL*	定义此区域的目标 URL。
nohref	nohref	从图像映射排除某个区域。
shape	`default` `rect` `circ` `poly`	定义区域的形状。
target	`_blank` `_parent` `_self` `_top`	规定在何处打开 href 属性指定的目标 URL。

全局属性

<area> 标签支持 HTML 中的全局属性。

事件属性

<area> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Area 对象

HTML <article> 标签

实例

```
<article>
  <h1>Internet Explorer 9</h1>
  <p>Windows Internet Explorer 9（简称 IE9）于 2011 年 3 月 14 日发布.
</article>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9, Firefox, Opera, Chrome 以及 Safari 支持 <article> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <article> 标签。

定义和用法

<article> 标签规定独立的自包含内容。

一篇文章应有其自身的意义，应该有可能独立于站点的其余部分对其进行分发。

<article> 元素的潜在来源：

- 论坛帖子
- 报纸文章
- 博客条目
- 用户评论

HTML 4.01 与 HTML 5 之间的差异

<article> 标签是 HTML 5 中的新标签。

全局属性

<article> 标签支持 HTML 中的全局属性。

事件属性

<article> 标签支持 HTML 中的事件属性。

HTML <aside> 标签

实例

```
<p>Me and my family visited The Epcot center this summer.</p>
<aside>
<h4>Epcot Center</h4>
The Epcot Center is a theme park in Disney World, Florida.
</aside>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <aside> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <aside> 标签。

定义和用法

<aside> 标签定义其所处内容之外的内容。

aside 的内容应该与附近的内容相关。

HTML 4.01 与 HTML 5 之间的差异

<aside> 标签是 HTML 5 的新标签。

提示和注释：

提示：<aside> 的内容可用作文章的侧栏。

全局属性

<aside> 标签支持 HTML 中的全局属性。

事件属性

<aside> 标签支持 HTML 中的事件属性。

HTML <audio> 标签

实例

一段简单的 HTML 5 音频：

```
<audio src="someaudio.wav">
您的浏览器不支持 audio 标签。
</audio>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <audio> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <audio> 标签。

定义和用法

<audio> 标签定义声音，比如音乐或其他音频流。

HTML 4.01 与 HTML 5 之间的差异

<audio> 标签是 HTML 5 的新标签。

提示和注释

提示：可以在开始标签和结束标签之间放置文本内容，这样老的浏览器就可以显示出不支持该标签的信息。

属性

new : HTML5 中的新属性。

属性	值	描述
autoplay	autoplay	如果出现该属性，则音频在就绪后马上播放。
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮。
loop	loop	如果出现该属性，则每当音频结束时重新开始播放。
muted	muted	规定视频输出应该被静音。
preload	preload	如果出现该属性，则音频在页面加载时进行加载，并预备播放。如果使用 "autoplay"，则忽略该属性。
src	*url*	要播放的音频的 URL。

全局属性

<audio> 标签支持 HTML 中的全局属性。

事件属性

<audio> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Audio 对象

HTML <b> 标签

实例

```
<p>这是普通文本 - <b>这是粗体文本</b>。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <b> 标签。

定义和用法

<b> 标签规定粗体文本。

HTML 4.01 与 HTML5 之间的差异

无。

提示和注释

注释：根据 HTML5 规范，在没有其他合适标签更合适时，才应该把 <b> 标签作为最后的选项。HTML5 规范声明：应该使用 <h1> - <h6> 来表示标题，使用 <em> 标签来表示强调的文本，应该使用 <strong> 标签来表示重要文本，应该使用 <mark> 标签来表示标注的/突出显示的文本。

提示：您也能够使用 CSS "font-weight" 属性来设置粗体文本。

全局属性

<b> 标签支持 HTML 中的全局属性。

事件属性

<b> 标签支持 HTML 中的事件属性。

HTML <base> 标签

实例

```
<head>
<base href="http://www.w3school.com.cn/i/" />
<base target="_blank" />
</head>

<body>
<img src="eg_smile.gif" />
<a href="http://www.w3school.com.cn">W3School</a>
</body>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <base> 标签。

定义和用法

<base> 标签为页面上的所有链接规定默认地址或默认目标。

通常情况下，浏览器会从当前文档的 URL 中提取相应的元素来填写相对 URL 中的空白。

使用 <base> 标签可以改变这一点。浏览器随后将不再使用当前文档的 URL，而使用指定的基本 URL 来解析所有的相对 URL。这其中包括 <a>、<img>、<link>、<form> 标签中的 URL。

HTML 与 XHTML 之间的差异

在 HTML 中，<base> 标签没有结束标签；在 XHTML 中，<base> 标签必须被正确地关闭。

提示和注释：

注释：<base> 标签必须位于 head 元素内部。

必需的属性

属性	值	描述
href	*URL*	规定页面中所有相对链接的基准 URL。

可选的属性

属性	值	描述
target	`_blank` `_parent` `_self` `_top` `framename`	在何处打开页面中所有的链接。

标准的属性和事件

NONE

相关页面

HTML DOM 参考手册：Base 对象

HTML <basefont> 标签

实例

规定页面上的默认字体颜色和字号：

```
<head>
<basefont color="red" size="5" />
</head>

<body>
<h1>This is a header</h1>
<p>This is a paragraph</p>
</body>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

只有 Internet Explorer 支持 <basefont> 标签。应该避免使用该标签。

定义和用法

<basefont> 标签定义基准字体。该标签可以为文档中的所有文本定义默认字体颜色、字体大小和字体系列。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，不赞成使用 basefont 元素；在 XHTML 1.0 Strict DTD 中，不支持 basefont 元素。

可选的属性

属性	值	描述
color	*rgb(x,x,x) #xxxxxx colorname*	不赞成使用。请使用样式取代它。规定文档中的默认文本颜色。
face	*font_family*	不赞成使用。请使用样式取代它。规定文档中的默认字体。
size	*number*	不赞成使用。请使用样式取代它。规定文档中的默认字体大小。

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
NONE
```

HTML <bdi> 标签

实例

把用户名从周围的文本方向设置中隔离出来：

```
<ul>
<li>Username <bdi>Bill</bdi>:80 points</li>
<li>Username <bdi>Steve</bdi>: 78 points</li>
</ul>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前只有 Firefox 和 Chrome 支持 <bdi> 标签。

定义和用法

bdi 指的是 bidi 隔离。

<bdi> 标签允许您设置一段文本，使其脱离其父元素的文本方向设置。

在发布用户评论或其他您无法完全控制的内容时，该标签很有用。

HTML 4.01 与 HTML 5 之间的差异

<bdi> 标签是 HTML5 中的新标签。

属性

属性	值	描述
dir	`ltr` `rtl` `auto`	可选。规定 <bdi> 元素内的文本的文本方向。默认值：auto。

全局属性

<bdi> 标签支持 HTML 中的全局属性。

事件属性

<bdi> 标签支持 HTML 中的事件属性。

HTML <bdo> 标签

实例

```
<bdo dir="rtl">Here is some text</bdo>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <bdo> 标签。

定义和用法

bdo 元素可覆盖默认的文本方向。

HTML 与 XHTML 之间的差异

NONE

可选的属性

属性	值	描述
dir	`ltr` `rtl`	定义文字的方向

全局属性

<bdo> 标签支持 HTML 中的全局属性。

事件属性

<bdo> 标签支持 HTML 中的事件属性。

TIY 实例

文字方向

此例演示如何改变文字的方向。

HTML <big> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <big> 标签。

定义和用法

<big> 标签呈现大号字体效果。

使用 <big> 标签可以很容易地放大字体。这简直不能再简单了：浏览器显示包含在 <big> 标签和其相应的 </big> 标签之间的文字时，其字体比周围的文字要大一号。但是，如果文字已经是最大号字体，这个 <big> 标签将不起任何作用。

更妙的是，可以嵌套 <big> 标签来放大文本。每一个 <big> 标签都可以使字体大一号，直到上限 7 号文本，正如字体模型所定义的那样。

但是使用 <big> 标签的时候还是要小心，因为浏览器总是很宽大地试图去理解各种标签，对于那些不支持 <big> 标签的浏览器来说，它经常将其认为是粗体字标签。

参阅

请参阅 HTML 5 参考手册，了解有关 <big> 标签的最新信息：

HTML 5 <big> 标签

HTML <blockquote> 标签

实例

标记长的引用：

```
<blockquote>
Here is a long quotation here is a long quotation here is a long qu
here is a long quotation here is a long quotation here is a long qu
here is a long quotation here is a long quotation here is a long qu
</blockquote>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流的浏览器均支持 <blockquote> 标签。

注释：没有浏览器能够正确地显示 cite 属性。

定义和用法

<blockquote> 标签定义块引用。

<blockquote> 与 </blockquote> 之间的所有文本都会从常规文本中分离出来，经常会在左、右两边进行缩进（增加外边距），而且有时会使用斜体。也就是说，块引用拥有它们自己的空间。

HTML 与 XHTML 之间的差异

没有。

提示和注释：

提示：请使用 q 元素来标记短的引用。

注释：如需把页面作为 strict XHTML 进行验证，那么 <blockquote> 元素必须包含块级元素，比如这样：

```
<blockquote>
<p>here is a long quotation here is a long quotation</p>
</blockquote>
```

可选的属性

属性	值	描述
cite	*URL*	规定引用的来源。

全局属性

<blockquote> 标签支持 HTML 中的全局属性。

事件属性

<blockquote> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Blockquote 对象

HTML <body> 标签

实例

一个简单的 HTML 文档，带有最基本的必需的元素：

```
<html>

<head>
  <title>文档的标题</title>
</head>

<body>
  文档的内容... ...
</body>

</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <body> 标签。

定义和用法

body 元素定义文档的主体。

body 元素包含文档的所有内容（比如文本、超链接、图像、表格和列表等等。）

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，所有 body 元素的“呈现属性”均不被赞成使用。

在 XHTML 1.0 Strict DTD 中，所有 body 元素的“呈现属性”均不被支持。

可选的属性

属性	值	描述
alink	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取代它。规定文档中活动链接（active link）的的颜色。
background	*URL*	不赞成使用。请使用样式取代它。规定文档的背景图像。
bgcolor	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取代它。规定文档的背景颜色。
link	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取代它。规定文档中未访问链接的默认颜色。
text	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取代它。规定文档中所有文本的颜色。
vlink	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取代它。规定文档中已被访问链接的颜色。

全局属性

<body> 标签支持 HTML 中的全局属性。

事件属性

<body> 标签支持 HTML 中的事件属性。

TIY 实例

一个简单的 HTML 文件

这个例子是一个很简单的 HTML 文件，使用了尽量少的 HTML 标签。它向您演示了 body 元素中的内容是如何被浏览器显示的。

```
<html>

<head>
<title>我的第一个 HTML 页面</title>
</head>

<body>
<p>body 元素的内容会显示在浏览器中。</p>
<p>title 元素的内容会显示在浏览器的标题栏中。</p>
</body>

</html>
```

背景颜色

本例演示如何为 HTML 页面添加背景颜色。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'
<meta http-equiv="Content-Language" content="zh-cn" />
</head>

<body bgcolor="yellow">
<h2>请看: 改变了颜色的背景。</h2>
</body>

</html>
```

背景图片

本例演示如何添加背景图片到 HTML 页面。

```
<html>

<body background="/i/eg_background.jpg">

<h3>图像背景</h3>

<p>gif 和 jpg 文件均可用作 HTML 背景。</p>

<p>如果图像小于页面，图像会进行重复。</p>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Column 对象

HTML
 标签

TIY 实例

换行

本例演示如何在文档中插入换行符。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持
 标签。

定义和用法

 可插入一个简单的换行符。

 标签是空标签（意味着它没有结束标签，因此这是错误的：
</br>）。在 XHTML 中，把结束标签放在开始标签中，也就是
。

请注意，
 标签只是简单地开始新的一行，而当浏览器遇到 <p> 标签时，通常会在相邻的段落之间插入一些垂直的间距。

clear 属性

如果您希望文本流在内联表格或图像的下一行继续输出，请使用 clear 属性，该属性有三个可选的值：left、right 或者 all，每个值都代表一个边界或两边的边界。

更多有关
 标签的 clear 属性的信息。

HTML 与 XHTML 之间的差异

在 HTML 中，
 标签没有结束标签。

在 XHTML 中，
 标签必须被正确地关闭，比如这样：
。

提示和注释：

注释：请使用
 来输入空行，而不是分割段落。

全局属性

 标签支持 HTML 中的全局属性。

事件属性

 标签支持 HTML 中的事件属性。

HTML <button> 标签

实例

以下代码标记了一个按钮：

```
<button type="button">Click Me!</button>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <button> 标签。

重要事项：如果在 HTML 表单中使用 button 元素，不同的浏览器会提交不同的值。Internet Explorer 将提交 <button> 与 <button/> 之间的文本，而其他浏览器将提交 value 属性的内容。请在 HTML 表单中使用 input 元素来创建按钮。

定义和用法

<button> 标签定义一个按钮。

在 button 元素内部，您可以放置内容，比如文本或图像。这是该元素与使用 input 元素创建的按钮之间的不同之处。

<button> 控件 与 <input type="button"> 相比，提供了更为强大的功能和更丰富的内容。<button> 与 </button> 标签之间的所有内容都是按钮的内容，其中包括任何可接受的正文内容，比如文本或多媒体内容。例如，我们可以在按钮中包括一个图像和相关的文本，用它们在按钮中创建一个吸引人的标记图像。

唯一禁止使用的元素是图像映射，因为它对鼠标和键盘敏感的动作会干扰表单按钮的行为。

请始终为按钮规定 type 属性。Internet Explorer 的默认类型是 "button"，而其他浏览器中（包括 W3C 规范）的默认值是 "submit"。

HTML 4.01 与 HTML 5 之间的差异

HTML 5 中的新属性：autofocus, form, formaction, formenctype, formmethod, formnovalidate 以及 formtarget。

提示和注释

注释：如果在 HTML 表单中使用 button 元素，不同的浏览器会提交不同的按钮值。请使用 input 元素在 HTML 表单中创建按钮。

属性

new : HTML5 中的新属性。

属性	值	描述
autofocus	autofocus	规定当页面加载时按钮应当自动地获得焦点。
disabled	disabled	规定应该禁用该按钮。
form	*form_name*	规定按钮属于一个或多个表单。
formaction	*url*	覆盖 form 元素的 action 属性。注释：该属性与 type="submit" 配合使用。
formenctype	见注释	覆盖 form 元素的 enctype 属性。注释：该属性与 type="submit" 配合使用。
formmethod	`get` `post`	覆盖 form 元素的 method 属性。注释：该属性与 type="submit" 配合使用。
formnovalidate	formnovalidate	覆盖 form 元素的 novalidate 属性。注释：该属性与 type="submit" 配合使用。
formtarget	`_blank` `_self` `_parent` `_top` `framename`	覆盖 form 元素的 target 属性。注释：该属性与 type="submit" 配合使用。
name	*button_name*	规定按钮的名称。
type	`button` `reset` `submit`	规定按钮的类型。
value	*text*	规定按钮的初始值。可由脚本进行修改。

注释：formenctype 属性可能的值：

- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

全局属性

<button> 标签支持 HTML 中的全局属性。

事件属性

<button> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Button 对象

HTML <canvas> 标签

实例

如何通过 canvas 元素来显示一个红色的矩形：

```
<canvas id="myCanvas"></canvas>

<script type="text/javascript">

var canvas=document.getElementById('myCanvas');
var ctx=canvas.getContext('2d');
ctx.fillStyle='#FF0000';
ctx.fillRect(0,0,80,100);

</script>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <canvas> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <canvas> 标签。

定义和用法

<canvas> 标签定义图形，比如图表和其他图像。

<canvas> 标签只是图形容器，您必须使用脚本来绘制图形。

HTML 4.01 与 HTML 5 之间的差异

<canvas> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
height	*pixels*	设置 canvas 的高度。
width	*pixels*	设置 canvas 的宽度。

全局属性

<canvas> 标签支持 HTML 中的全局属性。

事件属性

<canvas> 标签支持 HTML 中的事件属性。

<canvas> 的历史

这个 HTML 元素是为了客户端矢量图形而设计的。它自己没有行为，但却把一个绘图 API 展现给客户端 JavaScript 以使脚本能够把想绘制的东西都绘制到一块画布上。

<canvas> 标记由 Apple 在 Safari 1.3 Web 浏览器中引入。对 HTML 的这一根本扩展的原因在于，HTML 在 Safari 中的绘图能力也为 Mac OS X 桌面的 Dashboard 组件所使用，并且 Apple 希望有一种方式在 Dashboard 中支持脚本化的图形。

Firefox 1.5 和 Opera 9 都跟随了 Safari 的引领。这两个浏览器都支持 <canvas> 标记。

我们甚至可以在 IE 中使用 <canvas> 标记，并在 IE 的 VML 支持的基础上用开源的 JavaScript 代码（由 Google 发起）来构建兼容性的画布。 参见：http://excanvas.sourceforge.net/。

<canvas> 的标准化的努力由一个 Web 浏览器厂商的非正式协会在推进，目前 <canvas> 已经成为 HTML 5 草案中一个正式的标签。 参见：http://www.whatwg.org/specs/web-apps/current-work/

<canvas> 标记和 SVG 以及 VML 之间的差异

<canvas> 标记和 SVG 以及 VML 之间的一个重要的不同是，<canvas> 有一个基于 JavaScript 的绘图 API，而 SVG 和 VML 使用一个 XML 文档来描述绘图。

这两种方式在功能上是等同的，任何一种都可以用另一种来模拟。从表面上看，它们很不相同，可是，每一种都有强项和弱点。例如，SVG 绘图很容易编辑，只要从其描述中移除元素就行。

要从同一图形的一个 <canvas> 标记中移除元素，往往需要擦掉绘图重新绘制它。

如何使用 <canvas> 标记绘图

大多数 Canvas 绘图 API 都没有定义在 <canvas> 元素本身上，而是定义在通过画布的 getContext() 方法获得的一个“绘图环境”对象上。

Canvas API 也使用了路径的表示法。但是，路径由一系列的方法调用来定义，而不是描述为字母和数字的字符串，比如调用 beginPath() 和 arc() 方法。

一旦定义了路径，其他的方法，如 fill()，都是对此路径操作。绘图环境的各种属性，比如 fillStyle，说明了这些操作如何使用。

注释：Canvas API 非常紧凑的一个原因上它没有对绘制文本提供任何支持。要把文本加入到一个 <canvas> 图形，必须要么自己绘制它再用位图图像合并它，或者在 <canvas> 上方使用 CSS 定位来覆盖 HTML 文本。

相关页面

HTML 教程：HTML <canvas> 元素

HTML DOM 参考手册：HTML DOM Canvas 对象

HTML <caption> 标签

实例

```
<table border="1">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <caption> 标签。

定义和用法

caption 元素定义表格标题。

caption 标签必须紧随 table 标签之后。您只能对每个表格定义一个标题。通常这个标题会被居中于表格之上。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，caption 元素的 align 属性是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，caption 元素的 align 属性是不被支持的。

可选的属性

属性	值	描述
align	`left` `right` `top` `bottom`	不赞成使用。请使用样式取而代之。规定标题的对齐方式。

全局属性

<caption> 标签支持 HTML 中的全局属性。

事件属性

<caption> 标签支持 HTML 中的事件属性。

HTML <center> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <center> 标签。

定义和用法

对其所包括的文本进行水平居中。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，center 元素不被赞成使用。

在 XHTML 1.0 Strict DTD 中，center 元素不被支持。

提示和注释：

注释：提示：请使用 CSS 样式来居中文本！

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

HTML <cite> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <cite> 标签。

定义和用法

<cite> 标签通常表示它所包含的文本对某个参考文献的引用，比如书籍或者杂志的标题。

按照惯例，引用的文本将以斜体显示。

用 <cite> 标签把指向其他文档的引用分离出来，尤其是分离那些传统媒体中的文档，如书籍、杂志、期刊，等等。如果引用的这些文档有联机版本，还应该把引用包括在一个 <a> 标签中，从而把一个超链接指向该联机版本。

<cite> 标签还有一个隐藏的功能：它可以使你或者其他人从文档中自动摘录参考书目。我们可以很容易地想象一个浏览器，它能够自动整理引用表格，并把它们作为脚注或者独立的文档来显示。<cite> 标签的语义已经远远超过了改变它所包含的文本外观的作用；它使浏览器能够以各种实用的方式来向用户表达文档的内容。

HTML <em> <strong> <dfn> <code> <samp> <kbd><var> <cite> 标签

定义和用法

以下元素都是短语元素。虽然这些标签定义的文本大多会呈现出特殊的样式，但实际上，这些标签都拥有确切的语义。

我们并不反对使用它们，但是如果您只是为了达到某种视觉效果而使用这些标签的话，我们建议您使用样式表，那么做会达到更加丰富的效果。

| <em> | 把文本定义为强调的内容。 | | <strong> | 把文本定义为语气更强的强调的内容。 | | <dfn> | 定义一个定义项目。 | | <code> | 定义计算机代码文本。 | | <samp> | 定义样本文本。 | | <kbd> | 定义键盘文本。它表示文本是从键盘上键入的。它经常用在与计算机相关的文档或手册中。 | | <var> | 定义变量。您可以将此标签与 <pre> 及 <code> 标签配合使用。 | | <cite> | 定义引用。可使用该标签对参考文献的引用进行定义，比如书籍或杂志的标题。 |

HTML 与 XHTML 之间的差异

无

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

HTML 文本格式化实例

文本格式化

此例演示如何在一个 HTML 文件中对文本进行格式化

```
<html>

<body>

<b>This text is bold</b>

<br />

<strong>This text is strong</strong>

<br />

<big>This text is big</big>

<br />

<em>This text is emphasized</em>

<br />

<i>This text is italic</i>

<br />

<small>This text is small</small>

<br />

This text contains
<sub>subscript</sub>

<br />

This text contains
<sup>superscript</sup>

</body>
</html>
```

“计算机输出”标签

此例演示不同的“计算机输出”标签的显示效果。

```
<html>

<body>

<code>Computer code</code>
<br />
<kbd>Keyboard input</kbd>
<br />
<tt>Teletype text</tt>
<br />
<samp>Sample text</samp>
<br />
<var>Computer variable</var>
<br />

<p>
<b>注释：</b>这些标签常用于显示计算机/编程代码。
</p>

</body>
</html>
```

HTML <col> 标签

实例

col 元素为表格中的三个列规定了不同的对齐方式：

```
<table width="100%" border="1">
  <col align="left" />
  <col align="left" />
  <col align="right" />
  <tr>
    <th>ISBN</th>
    <th>Title</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>3476896</td>
    <td>My first HTML</td>
    <td>$53</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <col> 标签。

定义和用法

<col> 标签为表格中一个或多个列定义属性值。

如需对全部列应用样式，<col> 标签很有用，这样就不需要对各个单元和各行重复应用样式了。

您只能在 table 或 colgroup 元素中使用 <col> 标签。

HTML 与 XHTML 之间的差异

在 HTML 中，<col> 没有结束标签。

在 XHTML 中，<col> 标签必须被正确的关闭。

提示和注释：

提示：请为 <col> 标签添加 class 属性。这样就可以使用 CSS 来负责对齐方式、宽度和颜色等等。

提示：如果您希望在 colgroup 内部为每个列规定不同的属性值时，请使用此元素。如果没有 col 元素，列会从 colgroup 那里继承所有的属性值。

注释：col 元素是仅包含属性的空元素。如需创建列，您就必须在 tr 元素内部规定 td 元素。

参见：<colgroup> 标签。

可选的属性

属性	值	描述
align	`right` `left` `center` `justify` `char`	规定与 col 元素相关的内容的水平对齐方式。
char	*character*	规定根据哪个字符来对齐与 col 元素相关的内容。
charoff	*number*	规定第一个对齐字符的偏移量。
span	*number*	规定 col 元素应该横跨的列数。
valign	`top` `middle` `bottom` `baseline`	定义与 col 元素相关的内容的垂直对齐方式。
width	*pixels % relative_length*	规定 col 元素的宽度。

全局属性

<col> 标签支持 HTML 中的全局属性。

事件属性

<col> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Column 对象

HTML <colgroup> 标签

实例

两个 colgroup 元素为表格中的三列规定了不同的对齐方式和样式（注意第一个 colgroup 元素横跨两列）：

```
<table width="100%" border="1">
  <colgroup span="2" align="left"></colgroup>
  <colgroup align="right" style="color:#0000FF;"></colgroup>
  <tr>
    <th>ISBN</th>
    <th>Title</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>3476896</td>
    <td>My first HTML</td>
    <td>$53</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <colgroup> 标签。

Firefox、Chrome 以及 Safari 仅支持 colgroup 元素的 span 和 width 属性。

定义和用法

<colgroup> 标签用于对表格中的列进行组合，以便对其进行格式化。

如需对全部列应用样式，<colgroup> 标签很有用，这样就不需要对各个单元和各行重复应用样式了。

<colgroup> 标签只能在 table 元素中使用。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

提示：请为 <colgroup> 标签添加 class 属性。这样就可以使用 CSS 来负责对齐方式、宽度和颜色等等。

注释：col 元素是仅包含属性的空元素。如需创建列，您就必须在 tr 元素内部规定 td 元素。

参见：<col> 标签。

可选的属性

属性	值	描述
align	`right` `left` `center` `justify` `char`	定义在列组合中内容的水平对齐方式。
char	*character*	规定根据哪个字符来对齐列组中的内容。
charoff	*number*	规定第一个对齐字符的偏移量。
span	*number*	规定列组应该横跨的列数。
valign	`top` `middle` `bottom` `baseline`	定义在列组合中内容的垂直对齐方式。
width	*pixels % relative_length*	规定列组合的宽度。

全局属性

<colgroup> 标签支持 HTML 中的全局属性。

事件属性

<colgroup> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Column 对象

HTML <command> 标签

实例

标记一个按钮：

```
<menu>
<command onclick="alert('Hello World')">
Click Me!</command>
</menu>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

没有浏览器支持 <command> 标签。

只有 Internet Explorer 9 （更早或更晚的版本都不支持）支持 <command> 标签。

定义和用法

command 元素表示用户能够调用的命令。

<command> 标签可以定义命令按钮，比如单选按钮、复选框或按钮。

只有当 command 元素位于 menu 元素内时， 该元素才是可见的。否则不会显示这个元素，但是可以用它规定键盘快捷键。

浏览器支持

目前只有 Internet Explorer 支持 <command> 标签。

HTML 4.01 与 HTML 5 之间的差异

<command> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
checked	checked	定义是否被选中。仅用于 radio 或 checkbox 类型。
disabled	disabled	定义 command 是否可用。
icon	*url*	定义作为 command 来显示的图像的 url。
label	*text*	为 command 定义可见的 label。
radiogroup	*groupname*	定义 command 所属的组名。仅在类型为 radio 时使用。
type	`checkbox` `command` `radio`	定义该 command 的类型。默认是 "command"。

全局属性

<command> 标签支持 HTML 中的全局属性。

事件属性

<command> 标签支持 HTML 中的事件属性。

HTML <datalist> 标签

实例

下面是一个 input 元素，datalist 中描述了其可能的值：

```
<input id="myCar" list="cars" />
<datalist id="cars">
  <option value="BMW">
  <option value="Ford">
  <option value="Volvo">
</datalist>
```

定义和用法

<datalist> 标签定义选项列表。请与 input 元素配合使用该元素，来定义 input 可能的值。

datalist 及其选项不会被显示出来，它仅仅是合法的输入值列表。

请使用 input 元素的 list 属性来绑定 datalist。

浏览器支持

所有主流浏览器都支持 <datalist> 标签，除了 Internet Explorer 和 Safari。

HTML 4.01 与 HTML 5 之间的差异

<datalist> 标签是 HTML 5 中的新标签。

全局属性

<datalist> 标签支持 HTML 中的全局属性。

事件属性

<datalist> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Datalist 对象

HTML <dd> 标签

实例

```
<dl>
   <dt>计算机</dt>
   <dd>用来计算的仪器 ... ...</dd>
   <dt>显示器</dt>
   <dd>以视觉方式显示信息的装置 ... ...</dd>
</dl>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <dd> 标签。

定义和用法

<dd> 在定义列表中定义条目的定义部分。

HTML 与 XHTML 之间的差异

NONE

全局属性

<dd> 标签支持 HTML 中的全局属性。

事件属性

<dd> 标签支持 HTML 中的事件属性。

TIY 实例

定义列表

本例演示一个定义列表。

```
<html>

<body>

<h2>一个定义列表：</h2>

<dl>
   <dt>计算机</dt>
   <dd>用来计算的仪器 ... ...</dd>
   <dt>显示器</dt>
   <dd>以视觉方式显示信息的装置 ... ...</dd>
</dl>

</body>
</html>
```

HTML <del> 标签

实例

```
a dozen is <del>20</del> 12 pieces
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <del> 标签。

定义和用法

定义文档中已被删除的文本。

提示和注释：

注释：请与 <ins> 标签配合使用，来描述文档中的更新和修正。

HTML 与 XHTML 之间的差异

NONE

可选的属性

属性	值	描述
cite	*URL*	指向另外一个文档的 URL，此文档可解释文本被删除的原因。
datetime	*YYYYMMDD*	定义文本被删除的日期和时间。

全局属性

<del> 标签支持 HTML 中的全局属性。

事件属性

<del> 标签支持 HTML 中的事件属性。

TIY 实例

删除和插入文本

此例演示如何标记文档中删除和插入的文本。

```
<html>

<body>

<p>一打有 <del>二十</del> <ins>十二</ins> 件。</p>

<p>大多数浏览器会改写为删除文本和下划线文本。</p>

<p>一些老式的浏览器会把删除文本和下划线文本显示为普通文本。</p>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Del 对象

HTML <details> 标签

实例

关于文档的细节：

```
<details>
<summary>Copyright 2011.</summary>
<p>All pages and graphics on this web site are the property of W3Sc
</details>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前只有 Chrome 和 Safari 6 支持 <details> 标签。

定义和用法

<details> 标签用于描述文档或文档某个部分的细节。

浏览器支持

目前只有 Chrome 支持 <details> 标签。

HTML 4.01 与 HTML 5 之间的差异

<details> 标签是 HTML 5 中的新标签。

提示和注释

提示：与 <summary> 标签 配合使用可以为 details 定义标题。标题是可见的，用户点击标题时，会显示出 details。

属性

new : HTML5 中的新属性。

属性	值	描述
open	open	定义 details 是否可见。

全局属性

<details> 标签支持 HTML 中的全局属性。

事件属性

<details> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Details 对象

HTML <dialog> 标签

实例

使用 <dialog> 元素：

```
<table border="1">
<tr>
  <th>一月 <dialog open>这是打开的对话窗口</dialog></th>
  <th>二月</th>
  <th>三月</th>
</tr>
<tr>
  <td>31</td>
  <td>28</td>
  <td>31</td>
</tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前只有 Chrome 和 Safari 6 支持 <dialog> 标签。

定义和用法

<dialog> 标签定义对话框或窗口。

HTML 4.01 与 HTML 5 之间的差异

<dialog> 标签是 HTML 5 的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
open	open	规定 dialog 元素是活动的，用户可与之交互。

全局属性

<dialog> 标签支持 HTML 中的全局属性。

事件属性

<dialog> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Dialog 对象

HTML <dir> 标签

不赞成使用 **dir** 元素！

实例

目录列表：

```
<dir>
   <li>HTML</li>
   <li>XHTML</li>
   <li>CSS</li>
</dir>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <dir> 标签。

定义和用法

<dir> 标签定义目录列表。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，dir 元素不被赞成使用。

在 XHTML 1.0 Strict DTD 中，dir 元素不被支持。

提示和注释：

提示：请使用 CSS 来为列表添加样式！在我们的 CSS 教程中，您可以找到更多有关为列表添加样式的细节。

可选的属性

属性	值	描述
compact	compact	不赞成使用。请使用样式代替。

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onfocus, onblur, onmousedown, onmouseup, onmou
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

HTML <div> 标签

实例

文档中的一个部分会显示为绿色：

```
<div style="color:#00FF00">
  <h3>This is a header</h3>
  <p>This is a paragraph.</p>
</div>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <div> 标签。

定义和用法

<div> 可定义文档中的分区或节（division/section）。

<div> 标签可以把文档分割为独立的、不同的部分。它可以用作严格的组织工具，并且不使用任何格式与其关联。

如果用 id 或 class 来标记 <div>，那么该标签的作用会变得更加有效。

用法

<div> 是一个块级元素。这意味着它的内容自动地开始一个新行。实际上，换行是 <div> 固有的唯一格式表现。可以通过 <div> 的 class 或 id 应用额外的样式。

不必为每一个 <div> 都加上类或 id，虽然这样做也有一定的好处。

可以对同一个 <div> 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种。这两者的主要差异是，class 用于元素组（类似的元素，或者可以理解为某一类元素），而 id 用于标识单独的唯一的元素。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，div 元素的 "align" 属性不被赞成使用。

在 XHTML 1.0 Strict DTD 中，div 元素的 "align" 属性不被支持。

提示和注释：

注释：<div> 是一个块级元素，也就是说，浏览器通常会在 div 元素前后放置一个换行符。

提示：请使用 <div> 元素来组合块级元素，这样就可以使用样式对它们进行格式化。

案例分析

```
<body>

 <h1>NEWS WEBSITE</h1>
  <p>some text. some text. some text...</p>
  ...

 <div class="news">
  <h2>News headline 1</h2>
  <p>some text. some text. some text...</p>
  ...
 </div>

 <div class="news">
  <h2>News headline 2</h2>
  <p>some text. some text. some text...</p>
  ...
 </div>

 ...
</body>
```

例子解释

正如您看到的，上面这段 HTML 模拟了新闻网站的结构。其中的每个 div 把每条新闻的标题和摘要组合在一起，也就是说，div 为文档添加了额外的结构。同时，由于这些 div 属于同一类元素，所以可以使用 class="news" 对这些 div 进行标识，这么做不仅为 div 添加了合适的语义，而且便于进一步使用样式对 div 进行格式化，可谓一举两得。

提示：如需更深入地学习 class 和 id 的用法，请阅《W3school 的结构化标记》这一章中 div、id 和其他帮手 这一节。

可选的属性

属性	值	描述
align	left right center justify	不赞成使用。请使用样式取而代之。规定 div 元素中的内容的对齐方式。

全局属性

<div> 标签支持 HTML 中的全局属性。

事件属性

<div> 标签支持 HTML 中的事件属性。

HTML <dl> 标签

实例

```
<dl>
   <dt>计算机</dt>
   <dd>用来计算的仪器 ... ...</dd>
   <dt>显示器</dt>
   <dd>以视觉方式显示信息的装置 ... ...</dd>
</dl>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <dl> 标签。

定义和用法

<dl> 标签定义了定义列表（definition list）。

<dl> 标签用于结合 <dt> （定义列表中的项目）和 <dd> （描述列表中的项目）。

HTML 与 XHTML 之间的差异

NONE

全局属性

<dl> 标签支持 HTML 中的全局属性。

事件属性

<dl> 标签支持 HTML 中的事件属性。

HTML <dt> 标签

实例

```
<dl>
   <dt>计算机</dt>
   <dd>用来计算的仪器 ... ...</dd>
   <dt>显示器</dt>
   <dd>以视觉方式显示信息的装置 ... ...</dd>
</dl>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <dt> 标签。

定义和用法

<dt> 标签定义了定义列表中的项目（即术语部分）。

HTML 与 XHTML 之间的差异

NONE

全局属性

<dt> 标签支持 HTML 中的全局属性。

事件属性

<dt> 标签支持 HTML 中的事件属性。

TIY 实例

定义列表

如何创建定义列表。

HTML <embed> 标签

定义和用法

<embed> 标签定义嵌入的内容，比如插件。

实例

```
<embed src="helloworld.swf" />
```

HTML 4.01 与 HTML 5 之间的差异

<embed> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
height	*pixels*	设置嵌入内容的高度。
src	*url*	嵌入内容的 URL。
type	*type*	定义嵌入内容的类型。
width	*pixels*	设置嵌入内容的宽度。

全局属性

<embed> 标签支持 HTML 中的全局属性。

事件属性

<embed> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Embed 对象

HTML <fieldset> 标签

实例

组合表单中的相关元素：

```
<form>
  <fieldset>
    <legend>health information</legend>
    height: <input type="text" />
    weight: <input type="text" />
  </fieldset>
</form>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <fieldset> 标签。

定义和用法

fieldset 元素可将表单内的相关元素分组。

<fieldset> 标签将表单内容的一部分打包，生成一组相关表单的字段。

当一组表单元素放到 <fieldset> 标签内时，浏览器会以特殊方式来显示它们，它们可能有特殊的边界、3D 效果，或者甚至可创建一个子表单来处理这些元素。

<fieldset> 标签没有必需的或唯一的属性。

<legend> 标签为 fieldset 元素定义标题。

HTML 4.01 与 HTML 5 之间的差异

HTML5 拥有如下属性：disabled、form、name，HTML 4.01 中不支持这些属性。

属性

属性	值	描述
disabled	disabled	规定应该禁用 fieldset。
form	*form_id*	规定 fieldset 所属的一个或多个表单。
name	*value*	规定 fieldset 的名称。

全局属性

<fieldset> 标签支持 HTML 中的全局属性。

事件属性

<fieldset> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Fieldset 对象

TIY 实例

围绕数据的 Fieldset

本例演示如何在数据周围绘制一个带标题的框。

```
<!DOCTYPE HTML>
<html>

<body>

<form>
  <fieldset>
    <legend>健康信息</legend>
    身高：<input type="text" />
    体重：<input type="text" />
  </fieldset>
</form>

<p>如果表单周围没有边框，说明您的浏览器太老了。</p>

</body>
</html>
```

HTML <figcaption> 标签

实例

用作文档中插图的图像，带有一个标题：：

```
<figure>
  <figcaption>黄浦江上的的卢浦大桥</figcaption>
  <img src="shanghai_lupu_bridge.jpg" width="350" height="234" />
</figure>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9, Firefox, Opera, Chrome 以及 Safari 支持 <figcaption> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <figcaption> 标签。

定义和用法

<figcaption> 标签定义 figure 元素的标题（caption）。

"figcaption" 元素应该被置于 "figure" 元素的第一个或最后一个子元素的位置。

浏览器支持

所有主流浏览器都支持 <figcaption> 标签。

HTML 4.01 与 HTML 5 之间的差异

<figcaption> 标签是 HTML 5 中的新标签。

全局属性

<figcaption> 标签支持 HTML 中的全局属性。

事件属性

<figcaption> 标签支持 HTML 中的事件属性。

HTML <figure> 标签

实例

用作文档中插图的图像：

```
<figure>
  <p>黄浦江上的的卢浦大桥</p>
  <img src="shanghai_lupu_bridge.jpg" width="350" height="234" />
</figure>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9, Firefox, Opera, Chrome 以及 Safari 支持 <figure> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <figure> 标签。

定义和用法

<figure> 标签规定独立的流内容（图像、图表、照片、代码等等）。

figure 元素的内容应该与主内容相关，但如果被删除，则不应对文档流产生影响。

浏览器支持

所有主流浏览器都支持 <figure> 标签。

HTML 4.01 与 HTML 5 之间的差异

<figure> 标签是 HTML 5 中的新标签。

提示和注释

提示：请使用 <figcaption> 元素为 figure 添加标题（caption）。

全局属性

<figure> 标签支持 HTML 中的全局属性。

事件属性

<figure> 标签支持 HTML 中的事件属性。

HTML <font> 标签

定义和用法

<font> 规定文本的字体、字体尺寸、字体颜色。

实例

规定文本字体、大小和颜色：

```
<font size="3" color="red">This is some text!</font>
<font size="2" color="blue">This is some text!</font>
<font face="verdana" color="green">This is some text!</font>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <font> 标签。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，font 元素不被赞成使用。

在 XHTML 1.0 Strict DTD 中，font 元素不被支持。

提示和注释：

提示：请使用样式（代替 <font>）来定义文本的字体、字体颜色、字体尺寸。

可选的属性

属性	值	描述
color	*rgb(x,x,x) #xxxxxx colorname*	不赞成使用。请使用样式取代它。规定文本的颜色。
face	*font_family*	不赞成使用。请使用样式取代它。规定文本的字体。
size	*number*	不赞成使用。请使用样式取代它。规定文本的大小。

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
NONE
```

如需完整的描述，请访问事件属性。

TIY 实例

设置文字的字体

本例演示如何设置文字的字体。

```
<html>
<body>
<h1 style="font-family:verdana">A heading</h1>
<p style="font-family:courier">A paragraph</p>
</body>
</html>
```

设置文字的尺寸

本例演示如何设置文字的尺寸。

```
<html>
<body>
<h1 style="font-size:150%">A heading</h1>
<p style="font-size:80%">A paragraph</p>
</body>
</html>
```

设置文字的颜色

本例演示如何设置文字的颜色。

```
<html>
<body>
<h1 style="color:blue">A heading</h1>
<p style="color:red">A paragraph</p>
</body>
</html>
```

设置文字的字体、字体尺寸、字体颜色

本例演示如何设置文字的字体、字体尺寸、字体颜色.

```
<html>
<body>
<p style="font-family:verdana;font-size:80%;color:green">
This is a paragraph with some text in it. This is a paragraph with
</p>
</body>
</html>
```

HTML <footer> 标签

实例

文档中的页脚部分：

```
<footer>
  <p>Posted by: W3School</p>
  <p>Contact information: <a href="mailto:someone@example.com">some
</footer>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9, Firefox, Opera, Chrome, and Safari 支持 <footer> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <footer> 标签。

定义和用法

<footer> 标签定义文档或节的页脚。

<footer> 元素应当含有其包含元素的信息。

页脚通常包含文档的作者、版权信息、使用条款链接、联系信息等等。

您可以在一个文档中使用多个 <footer> 元素。

HTML 4.01 与 HTML 5 之间的差异

<footer> 标签是 HTML 5 中的新标签。

提示和注释

提示：<footer> 元素内的联系信息应该位于 <address> 标签中。

全局属性

<footer> 标签支持 HTML 中的全局属性。

事件属性

<footer> 标签支持 HTML 中的事件属性。

HTML <form> 标签

例子

```
<form action="form_action.asp" method="get">
  <p>First name: <input type="text" name="fname" /></p>
  <p>Last name: <input type="text" name="lname" /></p>
  <input type="submit" value="Submit" />
</form>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <form> 标签。

定义和用法

<form> 标签用于为用户输入创建 HTML 表单。

表单能够包含 input 元素，比如文本字段、复选框、单选框、提交按钮等等。

表单还可以包含 menus、textarea、fieldset、legend 和 label 元素。

表单用于向服务器传输数据。

提示和注释

注释：form 元素是块级元素，其前后会产生折行。

HTML 与 XHTML 之间的差异

NONE

属性

new : HTML5 中的新属性。

属性	值	描述
accept	*MIME_type*	HTML 5 中不支持。
accept-charset	*charset_list*	规定服务器可处理的表单数据字符集。
action	*URL*	规定当提交表单时向何处发送表单数据。
autocomplete	`on` `off`	规定是否启用表单的自动完成功能。
enctype	见说明	规定在发送表单数据之前如何对其进行编码。
method	`get` `post`	规定用于发送 form-data 的 HTTP 方法。
name	*form_name*	规定表单的名称。
novalidate	novalidate	如果使用该属性，则提交表单时不进行验证。
target	`_blank` `_self` `_parent` `_top` `framename`	规定在何处打开 action URL。

说明

enctype 属性可能的值：

- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

全局属性

<form> 标签支持 HTML 中的全局属性。

事件属性

<form> 标签支持 HTML 中的事件属性。

TIY 实例

文本域(Text fields)

本例演示如何在HTML页面创建文本域。用户可以在文本域写入文本。

```
<html>

<body>

<form>
名：
<input type="text" name="firstname">
<br />
姓：
<input type="text" name="lastname">
</form>

</body>
</html>
```

密码域

本例演示如何创建HTML的密码域。

```
<html>

<body>

<form>
用户：
<input type="text" name="user">
<br />
密码：
<input type="password" name="password">
</form>
<p>
请注意，当您在密码域中键入字符时，浏览器将使用项目符号来代替这些字符。
</p>
</body>
</html>
```

复选框

本例演示如何在HTML页中创建文本框。用户可以选中或取消选取复选框。

```
<html>

<body>

<form>
我喜欢自行车：
<input type="checkbox" name="Bike">
<br />
我喜欢汽车：
<input type="checkbox" name="Car">
</form>

</body>
</html>
```

单选按钮

本例演示如何在HTML中创建单选按钮。

```
<html>

<body>

<form>
男性：
<input type="radio" checked="checked" name="Sex" value="male" />
<br />
女性：
<input type="radio" name="Sex" value="female" />
</form>

<p>当用户点击一个单选按钮时，该按钮会变为选中状态，其他所有按钮会变为非选中状态

</body>
</html>
```

文本域(Textarea)

本例演示如何创建一个文本域（多行文本输入控制）。用户可以在文本域中写入文本。在文本域中，可写入的字符字数不受限制。

```
<html>
<body>

<p>
This example cannot be edited
because our editor uses a textarea
for input,
and your browser does not allow
a textarea inside a textarea.
</p>

<textarea rows="10" cols="30">
The cat was playing in the garden.
```

创建按钮

本例演示如何创建按钮。你可以对按钮上的文字进行自定义。

```
<html>

<body>

<form>
<input type="button" value="Hello world!">
</form>

</body>
</html>
```

带有输入框和确认按钮的表单

本例演示如何向页面添加表单。此表单包含两个输入框和一个确认按钮。

```
<html>
<body>

<form action="/example/html/form_action.asp" method="get">
  <p>First name: <input type="text" name="fname" /></p>
  <p>Last name: <input type="text" name="lname" /></p>
  <input type="submit" value="Submit" />
</form>

<p>请单击确认按钮，输入会发送到服务器上名为 "form_action.asp" 的页面。</p>

</body>
</html>
```

带有复选框的表单

此表单包含两个复选框和一个确认按钮。

```
<html>

<body>

<form name="input" action="/html/html_form_action.asp" method="get'
I have a bike:
<input type="checkbox" name="vehicle" value="Bike" checked="checkec
<br />
I have a car:
<input type="checkbox" name="vehicle" value="Car" />
<br />
I have an airplane:
<input type="checkbox" name="vehicle" value="Airplane" />
<br /><br />
<input type="submit" value="Submit" />
</form>

<p>如果您点击 "Submit" 按钮，您将把输入传送到名为 html_form_action.asp 的

</body>
</html>
```

带有单选按钮的表单

此表单包含两个单选框和一个确认按钮。

```
<html>

<body>

<form name="input" action="/html/html_form_action.asp" method="get'
Male:
<input type="radio" name="Sex" value="Male" checked="checked">
<br />
Female:
<input type="radio" name="Sex" value="Female">
<br />
<input type ="submit" value ="Submit">
</form>

<p>如果您点击 "Submit" 按钮，您将把输入传送到名为 html_form_action.asp 的

</body>
</html>
```

从表单发送电子邮件

此例演示如何从表单发送电子邮件。

```
<html>

<body>
<form action="MAILTO:someone@w3school.com.cn" method="post" enctype

<h3>这个表单会把电子邮件发送到 W3School。</h3>
姓名：<br />
<input type="text" name="name" value="yourname" size="20">
<br />
电邮：<br />
<input type="text" name="mail" value="yourmail" size="20">
<br />
内容：<br />
<input type="text" name="comment" value="yourcomment" size="40">
<br /><br />
<input type="submit" value="发送">
<input type="reset" value="重置">

</form>
</body>
</html>
```

相关页面

HTML DOM 参考手册：Form 对象

HTML <frame> 标签

实例

简单的三框架页面：

```
<html>

<frameset cols="25%,50%,25%">
  <frame src="frame_a.htm" />
  <frame src="frame_b.htm" />
  <frame src="frame_c.htm" />
</frameset>

</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <frame> 标签。

定义和用法

<frame> 标签定义 frameset 中的一个特定的窗口（框架）。

frameset 中的每个框架都可以设置不同的属性，比如 border、scrolling、noresize 等等。

HTML 与 XHTML 之间的差异

在 HTML 中，<frame> 标签没有结束标签。

在 XHTML 中，<frame> 标签必须被正确地关闭。

提示和注释：

注释：如果您希望验证包含框架的页面，请确保 doctype 被设置为 "Frameset DTD"。阅读更多有关 DOCTYPE 的内容。

重要事项：您不能与 <frameset></frameset> 标签一起使用 <body></body> 标签。不过，如果您需要为不支持框架的浏览器添加一个 <noframes> 标签，请务必将此标签放置在 <body></body> 标签中！

可选的属性

属性	值	描述
frameborder	`0` `1`	规定是否显示框架周围的边框。
longdesc	*URL*	规定一个包含有关框架内容的长描述的页面。
marginheight	*pixels*	定义框架的上方和下方的边距。
marginwidth	*pixels*	定义框架的左侧和右侧的边距。
name	*name*	规定框架的名称。
noresize	noresize	规定无法调整框架的大小。
scrolling	`yes` `no` `auto`	规定是否在框架中显示滚动条。
src	*URL*	规定在框架中显示的文档的 URL。

标准属性

```
id, class, title, style
```

如需完整的描述，请访问标准属性。

TIY 实例

垂直框架

本例演示：如何使用三份不同的文档制作一个垂直框架。

```
<html>

<frameset cols="25%,50%,25%">

  <frame src="/example/html/frame_a.html">
  <frame src="/example/html/frame_b.html">
  <frame src="/example/html/frame_c.html">

</frameset>

</html>
```

水平框架

本例演示：如何使用三份不同的文档制作一个水平框架。

```
<html>

<frameset rows="25%,50%,25%">

  <frame src="/example/html/frame_a.html">
  <frame src="/example/html/frame_b.html">
  <frame src="/example/html/frame_c.html">

</frameset>

</html>
```

如何使用 <noframes> 标签

本例演示：如何使用 <noframes> 标签。

```
<html>

<frameset cols="25%,50%,25%">
  <frame src="/example/html/frame_a.html">
  <frame src="/example/html/frame_b.html">
  <frame src="/example/html/frame_c.html">

<noframes>
<body>您的浏览器无法处理框架！</body>
</noframes>

</frameset>

</html>
```

混合框架结构

本例演示如何制作含有三份文档的框架结构，同时将他们混合置于行和列之中。

```
<html>

<frameset rows="50%,50%">

<frame src="/example/html/frame_a.html">

<frameset cols="25%,75%">
<frame src="/example/html/frame_b.html">
<frame src="/example/html/frame_c.html">
</frameset>

</frameset>

</html>
```

含有 noresize="noresize" 属性的框架结构

本例演示 noresize 属性。在本例中，框架是不可调整尺寸的。在框架间的边框上拖动鼠标，你会发现边框是无法移动的。

```
<html>

<frameset cols="50%,*,25%">
  <frame src="/example/html/frame_a.html" noresize="noresize" />
  <frame src="/example/html/frame_b.html" />
  <frame src="/example/html/frame_c.html" />
</frameset>

</html>
```

导航框架

本例演示如何制作导航框架。导航框架包含一个将第二个框架作为目标的链接列表。名为 "contents.htm" 的文件包含三个链接。

```
<html>

<frameset cols="120,*">

  <frame src="/example/html/html_contents.html">
  <frame src="/example/html/frame_a.html" name="showframe">

</frameset>

</html>
```

内联框架

本例演示如何创建内联框架（HTML 页中的框架）。

```
<html>

<body>

<iframe src="/i/eg_landscape.jpg"></iframe>

<p>一些老的浏览器不支持 iframe。</p>
<p>如果得不到支持，iframe 是不可见的。</p>

</body>
</html>
```

跳转至框架内的一个指定的节

本例演示两个框架。其中的一个框架设置了指向另一个文件内指定的节的链接。这个 "link.htm" 文件内指定的节使用 <a name="C10"> 进行标识。

```
<html>

<frameset cols="20%,80%">

 <frame src="/example/html/frame_a.html">
 <frame src="/example/html/link.html#C10">

</frameset>

</html>
```

使用框架导航跳转至指定的节

本利演示两个框架。左侧的导航框架包含了一个链接列表，这些链接将第二个框架作为目标。第二个框架显示被链接的文档。导航框架其中的链接指向目标文件中指定的节。

```
<html>

<frameset cols="180,*">

<frame src="/example/html/content.html">
<frame src="/example/html/link.html" name="showframe">

</frameset>

</html>
```

HTML <frameset> 标签

实例

简单的三框架页面：

```
<html>

<frameset cols="25%,50%,25%">
  <frame src="frame_a.htm" />
  <frame src="frame_b.htm" />
  <frame src="frame_c.htm" />
</frameset>

</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <frameset> 标签。

定义和用法

frameset 元素可定义一个框架集。它被用来组织多个窗口（框架）。每个框架存有独立的文档。在其最简单的应用中，frameset 元素仅仅会规定在框架集中存在多少列或多少行。您必须使用 cols 或 rows 属性。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：如果您希望验证某个包含框架的页面，请确保 DTD 被设置为 "Frameset DTD"。阅读更多有关 XHTML 验证的内容。

重要事项：您不能与 <frameset></frameset> 标签一起使用 <body></body> 标签。不过，如果您需要为不支持框架的浏览器添加一个 <noframes> 标签，请务必将此标签放置在 <body></body> 标签中！

可选的属性

属性	值	描述
cols	`pixels` `%` `*`	定义框架集中列的数目和尺寸。有关 cols 属性的详细信息。
rows	`pixels` `%` `*`	定义框架集中行的数目和尺寸。有关 rows 属性的详细信息。

标准属性

```
id, class, title, style
```

如需完整的描述，请访问标准属性。

事件属性

```
NONE
```

如需完整的描述，请访问事件属性。

HTML <h1> 到 <h6> 标签

实例

六个不同的 HTML 标题：

```
<h1>这是标题 1</h1>
<h2>这是标题 2</h2>
<h3>这是标题 3</h3>
<h4>这是标题 4</h4>
<h5>这是标题 5</h5>
<h6>这是标题 6</h6>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <h1> - <h6> 标签。

定义和用法

<h1> - <h6> 标签可定义标题。<h1> 定义最大的标题。<h6> 定义最小的标题。

由于 h 元素拥有确切的语义，因此请您慎重地选择恰当的标签层级来构建文档的结构。因此，请不要利用标题标签来改变同一行中的字体大小。相反，我们应当使用层叠样式表定义来达到漂亮的显示效果。

如果您希望了解更多有关 HTML 标签选择和使用的信息，请阅读《Web 品质》。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，h1 - h6 元素的 "align" 属性不被推荐使用。

在 XHTML 1.0 Strict DTD 中，h1 - h6 元素的 "align" 属性不被支持。

可选的属性

属性	值	描述
align	left center right justify	不推荐使用。请使用样式替代它。规定标题中文本的排列。

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

TIY 实例

标题

本例演示在 HTML 文档中显示标题的标签。

```
<html>

<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

<p>请仅仅把标题标签用于标题文本。不要仅仅为了产生粗体文本而使用它们。请使用其它

</body>
</html>
```

居中排列的标题

本例演示一个居中排列的标题。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<body>

<h1 align="center">This is heading 1</h1>

<p>上面的标题在页面中进行了居中排列。上面的标题在页面中进行了居中排列。上面的标

</body>
</html>
```

HTML <head> 标签

实例

一个简单的 HTML 文档，带有最基本的必需的元素：

```
<html>

<head>
  <title>文档的标题</title>
</head>

<body>
  文档的内容... ...
</body>

</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <head> 标签。

定义和用法

<head> 标签用于定义文档的头部，它是所有头部元素的容器。<head> 中的元素可以引用脚本、指示浏览器在哪里找到样式表、提供元信息等等。

文档的头部描述了文档的各种属性和信息，包括文档的标题、在 Web 中的位置以及和其他文档的关系等。绝大多数文档头部包含的数据都不会真正作为内容显示给读者。

下面这些标签可用在 head 部分：<base>, <link>, <meta>, <script>, <style>, 以及 <title>。

<title> 定义文档的标题，它是 head 部分中唯一必需的元素。

提示和注释

提示：应该把 <head> 标签放在文档的开始处，紧跟在 <html> 后面，并处于 <body> 标签或 <frameset> 标签之前。

提示：请记住始终为文档规定标题！

HTML 与 XHTML 之间的差异

NONE

可选的属性

属性	值	描述
profile	*URL*	一个由空格分隔的 URL 列表，这些 URL 包含着有关页面的元数据信息。

profile 属性的更多信息

文档的头部经常会包含一些 <meta> 标签，用来告诉浏览器关于文档的附加信息。在将来，创作者可能会利用预先定义好的标准文档的元数据配置文件（metadata profile），以便更好地描述它们的文档。profile 属性提供了与当前文档相关联的配置文件的 URL。

配置文件的格式以及浏览器使用它们的方式都还没有进行定义，这个属性主要是为将来的开发而保留的占位符。

全局属性

<head> 标签支持 HTML 中的全局属性。

TIY 实例

文档的标题

头元素内部的标题信息不会被显示在浏览器窗口中。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'
<meta http-equiv="Content-Language" content="zh-cn" />

<title>标题不会显示在文档区</title>
</head>

<body>
<p>这段文本会显示出来。</p>
</body>

</html>
```

一个 target，所有的链接

本例显示如何使用 base 标签使页面中的所有标签在新窗口中打开。

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'
<meta http-equiv="Content-Language" content="zh-cn" />

<base target="_blank" />
</head>

<body>

<p>
<a href="http://www.w3school.com.cn" target="_blank">这个连接</a> 将
</p>

<p>
<a href="http://www.w3school.com.cn">这个连接</a> 也将在新窗口中加载，即
</p>

</body>
</html>
```

HTML 中的样式

本例演示如何使用添加到 <head> 部分的样式信息对 HTML 进行格式化。

```
<html>

<head>
<style type="text/css">
h1 {color: red}
p {color: blue}
</style>
</head>

<body>
<h1>header 1</h1>
<p>A paragraph.</p>
</body>

</html>
```

链接到一个外部样式表

本例演示如何通过 <link> 标签链接到一个外部样式表。

```
<html>

<head>
<link rel="stylesheet" type="text/css" href="/html/csstest1.css" >
</head>

<body>
<h1>我通过外部样式表进行格式化。</h1>
<p>我也一样！</p>
</body>

</html>
```

文档描述

Meta 元素中的信息可以描述 HTML 文档。

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'

<meta name="author"
content="w3school.com.cn">

<meta name="revised"
content="David Yang,8/1/07">

<meta name="generator"
content="Dreamweaver 8.0en">

</head>

<body>
<p>本文档的 meta 属性标识了创作者和编辑软件。</p>
</body>

</html>
```

文档关键字

Meta 元素中的信息可以描述文档的关键词。

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'

<meta name="description"
content="HTML examples">

<meta name="keywords"
content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript">

</head>

<body>
<p>本文档的 meta 属性描述了该文档和它的关键词。</p>
</body>

</html>
```

重定向

这个例子演示：在网址已经变更的情况下，将用户重定向到另外一个地址。

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'
<meta http-equiv="Refresh" content="5;url=http://www.w3school.com.c
</head>

<body>
<p>
对不起。我们已经搬家了。您的 URL 是 <a href="http://www.w3school.com.cn'
</p>

<p>您将在 5 秒内被重定向到新的地址。</p>

<p>如果超过 5 秒后您仍然看到本消息，请点击上面的链接。</p>

</body>
</html>
```

HTML <header> 标签

实例

对主页的介绍：

```
<header>
<h1>Welcome to my homepage</h1>
<p>My name is Donald Duck</p>
</header>

<p>The rest of my home page...</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <header> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <header> 标签。

定义和用法

<header> 标签定义文档的页眉（介绍信息）。

HTML 4.01 与 HTML 5 之间的差异

<header> 标签是 HTML 5 中的新标签。

全局属性

<header> 标签支持 HTML 中的全局属性。

事件属性

<header> 标签支持 HTML 中的事件属性。

HTML <hr> 标签

实例

被水平线分隔的标题和段落：

```
<h1>This is header 1</h1>
<hr />
<p>This is some text</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <hr> 标签。

定义和用法

<hr> 标签在 HTML 页面中创建一条水平线。

水平分隔线（horizontal rule）可以在视觉上将文档分隔成各个部分。

HTML 与 XHTML 之间的差异

在 HTML 中，<hr> 标签没有结束标签。

在 XHTML 中，<hr> 必须被正确地关闭，比如 <hr />。

在 HTML 4.01 中，hr 元素的所有呈现属性均不被赞成使用。

在 XHTML 1.0 Strict DTD 中，hr 元素的所有呈现属性均不被支持。

可选的属性

属性	值	描述
align	`center` `left` `right`	不赞成使用。请使用样式取代它。规定 hr 元素的对齐方式。
noshade	noshade	不赞成使用。请使用样式取代它。规定 hr 元素的颜色呈现为纯色。
size	*pixels*	不赞成使用。请使用样式取代它。规定 hr 元素的高度（厚度）。
width	*pixels %*	不赞成使用。请使用样式取代它。规定 hr 元素的宽度。

全局属性

<hr> 标签支持 HTML 中的全局属性。

事件属性

<hr> 标签支持 HTML 中的事件属性。

TIY 实例

水平线

如何插入水平线。

```
<html>

<body>
<p>hr 标签定义水平线：</p>
<hr />
<p>这是段落。</p>
<hr />
<p>这是段落。</p>
<hr />
<p>这是段落。</p>
</body>
</html>
```

HTML <html> 标签

实例

```
<html>

<head>
  这里是文档的头部 ... ...
  ...
</head>

<body>
  这里是文档的主体 ... ...
  ...
</body>

</html>
```

定义和用法

此元素可告知浏览器其自身是一个 HTML 文档。

<html> 与 </html> 标签限定了文档的开始点和结束点，在它们之间是文档的头部和主体。正如您所了解的那样，文档的头部由 <head> 标签定义，而主体由 <body> 标签定义。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <html> 标签。

HTML 与 XHTML 之间的差异

xmlns 属性在 XHTML 中是必需的，但在 HTML 中不是。不过，即使 XHTML 文档中的 <html> 没有使用此属性，W3C 的验证器也不会报错。这是因为 "xmlns=http://www.w3.org/1999/xhtml" 是一个固定值，即使您没有包含它，此值也会被添加到 <html> 标签中。

有关 xmlns 属性的更多信息。

提示和注释

注释：即使 html 元素是文档的根元素，它也不包含 doctype 元素。doctype 元素必须位于 html 元素之前。

属性

new : HTML5 中的新属性。

属性	值	描述
manifest	*url*	定义一个 URL，在这个 URL 上描述了文档的缓存信息。
xmlns	http://www.w3.org/1999/xhtml	定义 XML namespace 属性。

全局属性

<html> 标签支持 HTML 中的全局属性。

HTML <i> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <i> 标签。

定义和用法

<i> 标签显示斜体文本效果。

<i> 标签和基于内容的样式标签 <em> 类似。它告诉浏览器将包含其中的文本以斜体字（italic）或者倾斜（oblique）字体显示。如果这种斜体字对该浏览器不可用的话，可以使用高亮、反白或加下划线等样式。

提示：<i> 标签一定要和结束标签 </i> 结合起来使用。

参阅

请参阅 HTML 5 参考手册，了解有关 <i> 标签的最新信息：

HTML 5 <i> 标签

HTML <iframe> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <iframe> 标签。

定义和用法

iframe 元素会创建包含另外一个文档的内联框架（即行内框架）。

HTML 与 XHTML 之间的差异

在 HTML 4.1 Strict DTD 和 XHTML 1.0 Strict DTD 中，不支持 iframe 元素。

提示和注释：

提示：您可以把需要的文本放置在 <iframe> 和 </iframe> 之间，这样就可以应对无法理解 iframe 的浏览器。

属性

new : HTML5 中的新属性。

属性	值	描述
align	`left` `right` `top` `middle` `bottom`	不赞成使用。请使用样式代替。规定如何根据周围的元素来对齐此框架。
frameborder	`1` `0`	规定是否显示框架周围的边框。
height	*pixels %*	规定 iframe 的高度。
longdesc	*URL*	规定一个页面，该页面包含了有关 iframe 的较长描述。
marginheight	*pixels*	定义 iframe 的顶部和底部的边距。
marginwidth	*pixels*	定义 iframe 的左侧和右侧的边距。
name	*frame_name*	规定 iframe 的名称。
sandbox	`""` `allow-forms` `allow-same-origin` `allow-scripts` `allow-top-navigation`	启用一系列对 <iframe> 中内容的额外限制。
scrolling	`yes` `no` `auto`	规定是否在 iframe 中显示滚动条。
seamless	seamless	规定 <iframe> 看上去像是包含文档的一部分。
src	*URL*	规定在 iframe 中显示的文档的 URL。
srcdoc	*HTML_code*	规定在 <iframe> 中显示的页面的 HTML 内容。
width	*pixels %*	定义 iframe 的宽度。

全局属性

<iframe> 标签支持 HTML 中的全局属性。

事件属性

<iframe> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：IFrame 对象

HTML <img> 标签

实例

在下面的例子中，我们在页面中插入一幅 W3School 的工程师在上海鲜花港拍摄的郁金香照片：

```
<img src="/i/eg_tulip.jpg"  alt="上海鲜花港 - 郁金香" />
```

以上代码的效果：



（您可以在页面底部找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <img> 标签。

定义和用法

img 元素向网页中嵌入一幅图像。

请注意，从技术上讲，<img> 标签并不会在网页中插入图像，而是从网页上链接图像。<img> 标签创建的是被引用图像的占位空间。

<img> 标签有两个必需的属性：src 属性 和 alt 属性。

延伸阅读：如何正确地使用图像

HTML 与 XHTML 之间的差异

在 HTML 中，<img> 标签没有结束标签。

在 XHTML 中，<img> 标签必须被正确地关闭。

在 HTML 4.01 中，不推荐使用 image 元素的 "align"、"border"、"hspace" 以及 "vspace" 属性。

在 XHTML 1.0 Strict DTD 中，不支持 image 元素的 "align"、"border"、"hspace" 以及 "vspace" 属性。

必需的属性

属性	值	描述
alt	*text*	规定图像的替代文本。
src	*URL*	规定显示图像的 URL。

可选的属性

属性	值	描述
align	`top` `bottom` `middle` `left` `right`	不推荐使用。规定如何根据周围的文本来排列图像。
border	*pixels*	不推荐使用。定义图像周围的边框。
height	*pixels %*	定义图像的高度。
hspace	*pixels*	不推荐使用。定义图像左侧和右侧的空白。
ismap	*URL*	将图像定义为服务器端图像映射。
longdesc	*URL*	指向包含长的图像描述文档的 URL。
usemap	*URL*	将图像定义为客户器端图像映射。
vspace	*pixels*	不推荐使用。定义图像顶部和底部的空白。
width	*pixels %*	设置图像的宽度。

全局属性

<img> 标签支持 HTML 中的全局属性。

事件属性

<img> 标签支持 HTML 中的事件属性。

TIY 实例

插入图像

本例演示如何在网页中显示图像。

```
<!DOCTYPE HTML>
<html>

<body>

<p>
一幅图像：
<img src="/i/eg_mouse.jpg" width="128" height="128" />
</p>

<p>
一幅动画图像：
<img src="/i/eg_cute.gif" width="50" height="50" />
</p>

<p>请注意，插入动画图像的语法与插入普通图像的语法没有区别。</p>

</body>
</html>
```

从不同的位置插入图片

本例演示如何将其他文件夹或服务器的图片显示到网页中。

```
<html>

<body>

<p>
来自另一个文件夹的图像：
<img src="/i/ct_netscape.jpg" />
</p>

<p>
来自 W3School.com.cn 的图像：
<img src="http://www.w3school.com.cn/i/w3school_logo_white.gif" />
</p>

</body>
</html>
```

背景图片

本例演示如何添加背景图片到HTML页面。

```
<html>

<body background="/i/eg_background.jpg">

<h3>图像背景</h3>

<p>gif 和 jpg 文件均可用作 HTML 背景。</p>

<p>如果图像小于页面，图像会进行重复。</p>

</body>
</html>
```

排列图片

本例演示如何在文字中排列图像。

```
<html>

<body>

<h2>未设置对齐方式的图像：</h2>

<p>图像 <img src ="/i/eg_cute.gif"> 在文本中</p>

<h2>已设置对齐方式的图像：</h2>

<p>图像 <img src="/i/eg_cute.gif" align="bottom"> 在文本中</p>

<p>图像 <img src ="/i/eg_cute.gif" align="middle"> 在文本中</p>

<p>图像 <img src ="/i/eg_cute.gif" align="top"> 在文本中</p>

<p>请注意，bottom 对齐方式是默认的对齐方式。</p>

</body>
</html>
```

浮动图像

本例演示如何使图片浮动至段落的左边或右边。

```
<html>

<body>

<p>
<img src ="/i/eg_cute.gif" align ="left">
带有图像的一个段落。图像的 align 属性设置为 "left"。图像将浮动到文本的左侧。
</p>

<p>
<img src ="/i/eg_cute.gif" align ="right">
带有图像的一个段落。图像的 align 属性设置为 "right"。图像将浮动到文本的右侧。
</p>

</body>
</html>
```

调整图像尺寸

本例演示如何将图片调整到不同的尺寸。

```
<html>

<body>

<img src="/i/eg_mouse.jpg" width="50" height="50">

<br />

<img src="/i/eg_mouse.jpg" width="100" height="100">

<br />

<img src="/i/eg_mouse.jpg" width="200" height="200">

<p>通过改变 img 标签的 "height" 和 "width" 属性的值，您可以放大或缩小图像。

</body>
</html>
```

为图片显示替换文本

本例演示如何为图片显示替换文本。在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息。为页面上的图像都加上替换文本属性是个好习惯。

```
<html>

<body>

<p>仅支持文本的浏览器无法显示图像，仅仅能够显示在图像的 "alt" 属性中指定的文本

<p>请注意，如果您把鼠标指针移动到图像上，大多数浏览器会显示 "alt" 文本。</p>

<img src="/i/eg_goleft.gif" alt="向左转" />

<p>如果无法显示图像，将显示 "alt" 属性中的文本：</p>

<img src="/i/eg_goleft123.gif" alt="向左转" />

</body>
</html>
```

制作图像链接

本例演示如何将图像作为一个链接使用。

```
<html>

<body>
<p>
您也可以把图像作为链接来使用：
<a href="/example/html/lastpage.html">
<img border="0" src="/i/eg_buttonnext.gif" />
</a>
</p>

</body>
</html>
```

创建图像地图

本例显示如何创建带有可供点击区域的图像地图。其中的每个区域都是一个超级链接。

```
<html>
<body>

<p>请点击图像上的星球，把它们放大。</p>

<img
src="/i/eg_planets.jpg"
border="0" usemap="#planetmap"
alt="Planets" />

<map name="planetmap" id="planetmap">

<area
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
target ="_blank"
alt="Venus" />

<area
shape="circle"
coords="129,161,10"
href ="/example/html/mercur.html"
target ="_blank"
alt="Mercury" />

<area
shape="rect"
coords="0,0,110,260"
href ="/example/html/sun.html"
target ="_blank"
alt="Sun" />

</map>

<p><b>注释：</b>img 元素中的 "usemap" 属性引用 map 元素中的 "id" 或 "nam

</body>
</html>
```

延伸阅读 - 如何正确地使用图像

HTML 和 XHTML 最引人注目的特征之一，就是能够在文档的文本中包括图像，既可以把图像作为文档的内在对象（内联图像），也可以将其作为一个可通过超链接下载的单独文档，或者作为文档的背景。

合理地在文档内容中加入图像（静态的或者具有动画效果的图标、照片、说明、绘画，等等）时，会使文档变得更加生动活泼，更加引人入胜，而且看上去更加专业，更具信息性并易于浏览。还可以专门使一个图像成为超链接的可视引导图。

然而，如果过度使用图像，文档就会变得支离破碎，混乱不堪，而且无法阅读，用户在下载和查看该页面时，更会增加很多不必要的等待时间。

请阅读下面的文章，学习 Web 上的两种主要图像格式：GIF 和 JPEG，以及如何正确地使用图像：

- GIF 图像
- JPEG 图像
- 在 Web 上使用图像

参阅

- 教程：HTML 图像
- 参考手册：HTML 5 <img> 标签

相关页面

HTML DOM 参考手册：Image 对象

HTML <input> 标签

实例

一个简单的 HTML 表单，包含两个文本输入框和一个提交按钮：

```
<form action="form_action.asp" method="get">
  First name: <input type="text" name="fname" />
  Last name: <input type="text" name="lname" />
  <input type="submit" value="Submit" />
</form>
```

（您可以在页面底部找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <input> 标签。

定义和用法

<input> 标签用于搜集用户信息。

根据不同的 type 属性值，输入字段拥有很多种形式。输入字段可以是文本字段、复选框、掩码后的文本控件、单选按钮、按钮等等。

HTML 与 XHTML 之间的差异

在 HTML 中，<input> 标签没有结束标签。

在 XHTML 中，<input> 标签必须被正确地关闭。

提示和注释：

提示：请使用 label 元素为某个表单控件定义标签。

属性

new : HTML5 中的新属性。

属性	值	描述
accept	*mime_type*	规定通过文件上传来提交的文件的类型。
align	`left` `right` `top` `middle` `bottom`	不赞成使用。规定图像输入的对齐方式。
alt	text	定义图像输入的替代文本。
autocomplete	`on` `off`	规定是否使用输入字段的自动完成功能。
autofocus	autofocus	规定输入字段在页面加载时是否获得焦点。（不适用于 type="hidden"）
checked	checked	规定此 input 元素首次加载时应当被选中。
disabled	disabled	当 input 元素加载时禁用此元素。
form	*formname*	规定输入字段所属的一个或多个表单。
formaction	*URL*	覆盖表单的 action 属性。（适用于 type="submit" 和 type="image"）
formenctype	见注释	覆盖表单的 enctype 属性。（适用于 type="submit" 和 type="image"）
formmethod	`get` `post`	覆盖表单的 method 属性。（适用于 type="submit" 和 type="image"）
formnovalidate	formnovalidate	覆盖表单的 novalidate 属性。如果使用该属性，则提交表单时不进行验证。
formtarget	`_blank` `_self` `_parent` `_top` `framename`	覆盖表单的 target 属性。（适用于 type="submit" 和 type="image"）
height	*pixels %*	定义 input 字段的高度。（适用于 type="image"）
list	*datalist-id*	引用包含输入字段的预定义选项的 datalist 。

max	*number date*	规定输入字段的最大值。请与 "min" 属性配合使用，来创建合法值的范围。
maxlength	number	规定输入字段中的字符的最大长度。
min	*number date*	规定输入字段的最小值。请与 "max" 属性配合使用，来创建合法值的范围。
multiple	multiple	如果使用该属性，则允许一个以上的值。
name	field_name	定义 input 元素的名称。
pattern	*regexp_pattern*	规定输入字段的值的模式或格式。例如 pattern="[0-9]" 表示输入值必须是 0 与 9 之间的数字。
placeholder	*text*	规定帮助用户填写输入字段的提示。
readonly	readonly	规定输入字段为只读。
required	required	指示输入字段的值是必需的。
size	number_of_char	定义输入字段的宽度。
src	URL	定义以提交按钮形式显示的图像的 URL。
step	*number*	规定输入字的的合法数字间隔。
type	`button` `checkbox` `file` `hidden` `image` `password` `radio` `reset` `submit` `text`	规定 input 元素的类型。
value	*value*	规定 input 元素的值。
width	*pixels %*	定义 input 字段的宽度。（适用于 type="image"）

全局属性

<input> 标签支持 HTML 中的全局属性。

事件属性

<input> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：

- Input Button 对象
- Input Checkbox 对象
- Input Color 对象
- Input Date 对象
- Input Datetime 对象
- Input Datetime Local 对象
- Input Email 对象
- Input FileUpload 对象
- Input Hidden 对象
- Input Input Image 对象
- Input Month 对象
- Input Number 对象
- Input Password 对象
- Input Range 对象
- Input Radio 对象
- Input Reset 对象
- Input Input Search 对象
- Input Submit 对象
- Input Text 对象
- Input Input Time 对象
- Input Input URL 对象
- Input Week 对象

HTML DOM Button 对象

Button 对象

Button 对象代表 HTML 文档中的一个按钮。

该元素没有默认的行为，但是必须有一个 onclick 事件句柄以便使用。

在 HTML 文档中 <input type="button"> 标签每出现一次，一个 Button 对象 就会被创建。

您可以通过遍历表单的 elements[] 数组来访问某个按钮，或者通过使用 document.getElementById()。

Button 对象的属性

属性	描述
accessKey	设置或返回访问按钮的快捷键。
alt	设置或返回当浏览器无法显示按钮时供显示的替代文本。
disabled	设置或返回是否禁用按钮。
form	返回对包含该按钮的表单对象的引用。
id	设置或返回按钮的 id。
name	设置或返回按钮的名称。
tabIndex	设置或返回按钮的 tab 键控制次序。
type	返回按钮的表单元素类型。
value	设置或返回在按钮上显示的文本。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Button 对象的方法

方法	描述
blur()	把焦点从元素上移开。
click()	在某个按钮上模拟一次鼠标单击。
focus()	为某个按钮赋予焦点。

HTML DOM Checkbox 对象

Checkbox 对象

Checkbox 对象代表一个 HTML 表单中的 一个选择框。

在 HTML 文档中 <input type="checkbox"> 每出现一次，Checkbox 对象就会被创建。

您可以通过遍历表单的 elements[] 数组来访问某个选择框，或者通过使用 document.getElementById() 。

Checkbox 对象的属性

属性	描述
accessKey	设置或返回访问 checkbox 的快捷键。
alt	设置或返回不支持 checkbox 时显示的替代文本。
checked	设置或返回 checkbox 是否应被选中。
defaultChecked	返回 checked 属性的默认值。
disabled	设置或返回 checkbox 是否应被禁用。
form	返回对包含 checkbox 的表单的引用。
id	设置或返回 checkbox 的 id。
name	设置或返回 checkbox 的名称。
tabIndex	设置或返回 checkbox 的 tab 键控制次序。
type	返回 checkbox 的表单元素类型。
value	设置或返回 checkbox 的 value 属性的值

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Checkbox 对象的方法

方法	描述
blur()	从 checkbox 上移开焦点。
click()	模拟在 checkbox 中的一次鼠标点击。
focus()	为 checkbox 赋予焦点。

HTML DOM Color 对象

Color 对象

Color 对象是 HTML5 中的新对象。

Color 对象表示 HTML <input type="color"> 元素。

注释：Internet Explorer 和 Safari 不支持 <input type="color"> 元素。

访问 Color 对象

您可以通过使用 getElementById() 来访问 <color> 元素：

```
var x = document.getElementById("myColor");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Color 对象。

创建 Color 对象

您可以通过使用 document.createElement() 方法来创建 <color> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "color");
```

Color 对象属性

属性	描述
span	设置或返回列的 span 属性的值。
autocomplete	设置或返回色彩选择器的 autocomplete 属性值。
autofocus	设置或返回色彩选择器在页面加载后是否应自动获取焦点。
defaultValue	设置或返回色彩选择器的默认值。
disabled	设置或返回色彩选择器是否被禁用。
form	返回对包含色彩选择器的表单的引用。
list	返回对包含色彩选择器的 datalist 的引用。
name	设置或返回色彩选择器的 name 属性值。
type	返回色彩选择器的表单元素类型。
value	设置或返回色彩选择器的 value 属性值。

标准属性和事件

Color 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Input Date 对象

Input Date 对象

Input Date 对象是 HTML5 中的新对象。

Input Date 对象表示 HTML <input type="datetime"> 元素。

注释：Internet Explorer 或 Firefox 不支持 <input type="date"> 元素。

访问 Input Date 对象

您可以通过使用 getElementById() 来访问 <input type="date"> 元素：

```
var x = document.getElementById("myDate");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Input Date 对象。

创建 Input Date 对象

您可以通过使用 document.createElement() 方法来创建 <input type="date"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "date");
```

Input Date 对象属性

属性	描述
autocomplete	设置或返回 date 字段的 autocomplete 属性值。
autofocus	设置或返回 date 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 date 字段的默认值。
disabled	设置或返回 date 字段是否被禁用。
form	返回对包含 date 字段的表单的引用。
list	返回对包含 date 字段的 datalist 的引用。
max	设置或返回 date 字段的 max 属性值。
min	设置或返回 date 字段的 min 属性值。
name	设置或返回 date 字段的 name 属性值。
readOnly	设置或返回 date 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 date 字段。
step	设置或返回 date 字段的 step 属性值。
type	返回 date 字段的表单元素类型。
value	设置或返回 date 字段的 value 属性值。

标准属性和事件

Input Date 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Datetime 对象

Datetime 对象

Datetime 对象是 HTML5 中的新对象。

Datetime 对象表示 HTML <input type="datetime"> 元素。

注释：Internet Explorer、Firefox 或者 Chrome 不支持 <input type="datetime"> 元素，Safari 中部分支持。Opera 12 以及更早的版本中完全支持。

访问 Datetime 对象

您可以通过使用 getElementById() 来访问 <input type="myDatetime"> 元素：

```
var x = document.getElementById("myDatetime");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Datetime 对象。

创建 Datetime 对象

您可以通过使用 document.createElement() 方法来创建 <input type="datetime"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "datetime");
```

Datetime 对象属性

属性	描述
autocomplete	设置或返回 datetime 字段的 autocomplete 属性值。
autofocus	设置或返回 datetime 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 datetime 字段的默认值。
disabled	设置或返回 datetime 字段是否被禁用。
form	返回对包含 datetime 字段的表单的引用。
list	返回对包含 datetime 字段的 datalist 的引用。
max	设置或返回 datetime 字段的 max 属性值。
min	设置或返回 datetime 字段的 min 属性值。
name	设置或返回 datetime 字段的 name 属性值。
readOnly	设置或返回 datetime 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 datetime 字段。
step	设置或返回 datetime 字段的 step 属性值。
type	返回 datetime 字段的表单元素类型。
value	设置或返回 datetime 字段的 value 属性值。

标准属性和事件

Datetime 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Datetime Local 对象

Datetime Local 对象

Datetime Local 对象是 HTML5 中的新对象。

Datetime Local 对象表示 HTML <colgroup> 元素。

注释：Internet Explorer 或 Firefox 不支持 <input type="datetime-local"> 元素。

访问 Datetime Local 对象

您可以通过使用 getElementById() 来访问 <datetime-local> 元素：

```
var x = document.getElementById("myLocalDate");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Input Date 对象。

创建 Datetime Local 对象

您可以通过使用 document.createElement() 方法来创建 <datetime-local> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "datetime-local");
```

Datetime Local 对象属性

属性	描述
autocomplete	设置或返回本地时间字段的 autocomplete 属性值。
autofocus	设置或返回本地时间字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回本地时间字段的默认值。
disabled	设置或返回本地时间字段是否被禁用。
form	返回对包含本地时间字段的表单的引用。
list	返回对包含本地时间字段的 datalist 的引用。
max	设置或返回本地时间字段的 max 属性值。
min	设置或返回本地时间字段的 min 属性值。
name	设置或返回本地时间字段的 name 属性值。
readOnly	设置或返回本地时间字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写本地时间字段。
step	设置或返回本地时间字段的 step 属性值。
type	返回本地时间字段的表单元素类型。
value	设置或返回本地时间字段的 value 属性值。

标准属性和事件

Datetime Local 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Email 对象

Email 对象

Email 对象是 HTML5 中的新对象。

Email 对象表示 HTML <email> 元素。

注释：Internet Explorer 9（以及更早的版本）或 Safari 不支持 <input type="email"> 元素。

访问 Email 对象

您可以通过使用 getElementById() 来访问 <email> 元素：

```
var x = document.getElementById("myEmail");
```

创建 Email 对象

您可以通过使用 document.createElement() 方法来创建 <email> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "email");
```

Email 对象属性

属性	描述
autocomplete	设置或返回 email 字段的 autocomplete 属性值。
autofocus	设置或返回 email 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 email 字段的默认值。
disabled	设置或返回 email 字段是否被禁用。
form	返回对包含 email 字段的表单的引用。
list	返回对包含 email 字段的 datalist 的引用。
max	设置或返回 email 字段的 max 属性值。
min	设置或返回 email 字段的 min 属性值。
name	设置或返回 email 字段的 name 属性值。
pattern	设置或返回 email 字段的 pattern 属性值。
placeholder	设置或返回 email 字段的 placeholder 属性值。
readOnly	设置或返回 email 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 email 字段。
size	设置或返回 email 字段的 size 属性值。
step	设置或返回 email 字段的 step 属性值。
type	返回 email 字段的表单元素类型。
value	设置或返回 email 字段的 value 属性值。

标准属性和事件

Email 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM FileUpload 对象

FileUpload 对象

在 HTML 文档中 <input type="file"> 标签每出现一次，一个 FileUpload 对象就会被创建。

该元素包含一个文本输入字段，用来输入文件名，还有一个按钮，用来打开文件选择对话框以便图形化选择文件。

该元素的 value 属性保存了用户指定的文件的名称，但是当包含一个 file-upload 元素的表单被提交的时候，浏览器会向服务器发送选中的文件的内容而不仅仅是发送文件名。

为安全起见，file-upload 元素不允许 HTML 作者或 JavaScript 程序员指定一个默认的文件名。HTML value 属性被忽略，并且对于此类元素来说，value 属性是只读的，这意味着只有用户可以输入一个文件名。当用户选择或编辑一个文件名，file-upload 元素触发 onchange 事件句柄。

您可以通过遍历表单的 elements[] 数组，或者通过使用 document.getElementById()来访问 FileUpload 对象。

FileUpload 对象的属性

属性	描述
accept	设置或返回指示文件传输的 MIME 类型的列表（逗号分隔）。
accessKey	设置或返回访问 FileUpload 对象的快捷键。
alt	设置或返回不支持 <input type="file"> 时显示的替代文字。
defaultValue	设置或返回 FileUpload 对象的初始值。
disabled	设置或返回是否禁用 FileUpload 对象。
form	返回对包含 FileUpload 对象的表单的引用。
id	设置或返回 FileUpload 对象的 id。
name	设置或返回 FileUpload 对象的名称。
tabIndex	设置或返回定义 FileUpload 对象的 tab 键控制次序的索引号。
type	返回表单元素的类型。对于 FileUpload ，则是 "file" 。
value	返回由用户输入设置的文本后，FileUpload 对象的文件名。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

FileUpload 对象的方法

方法	描述
blur()	从 FileUpload 对象上移开焦点。
focus()	为 FileUpload 对象赋予焦点。
select()	选取 FileUpload 对象。

HTML DOM Hidden 对象

Hidden 对象

Hidden 对象代表一个 HTML 表单中的某个隐藏输入域。

这种类型的输入元素实际上是隐藏的。这个不可见的表单元素的 value 属性保存了一个要提交给 Web 服务器的任意字符串。如果想要提交并非用户直接输入的数据的话，就是用这种类型的元素。

在 HTML 表单中 <input type="hidden"> 标签每出现一次，一个 Hidden 对象就会被创建。

您可通过遍历表单的 elements[] 数组来访问某个隐藏输入域，或者通过使用 document.getElementById()。

Hidden 对象的属性

属性	描述
alt	设置或返回当不支持隐藏输入域时显示的替代文本。
form	返回一个对包含隐藏域的表单的引用。
id	设置或返回隐藏域的 id。
name	设置或返回隐藏域的名称。
type	返回隐藏输入域的表单类型。
value	设置或返回隐藏域的 value 属性的值。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

HTML DOM Input Image 对象

Input Image 对象

Input Image 对象是 HTML5 中的新对象。

Input Image 对象表示 HTML <image> 元素。

访问 Input Image 对象

您可以通过使用 getElementById() 来访问 <image> 元素：

```
var x = document.getElementById("myImage");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Input Image 对象。

创建 Input Image 对象

您可以通过使用 document.createElement() 方法来创建 <image> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "image");
```

Input Image 对象属性

属性	描述
autocomplete	设置或返回 email 字段的 autocomplete 属性值。
alt	设置或返回 input image 的 alt 属性值。
autofocus	设置或返回 input image 在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 input image 的默认值。
disabled	设置或返回 input image 是否被禁用。
form	返回对包含 input image 的表单的引用。
formAction	设置或返回 input image 的 formaction 属性值。
formEnctype	设置或返回 input image 的 formenctype 属性值。
formMethod	设置或返回 input image 的 formmethod 属性值。
formNoValidate	设置或返回在提交表单时是否应该验证表单数据（form-data）。
formTarget	设置或返回 input image 的 formtarget 属性值。
height	设置或返回 input image 的 height 属性值。
name	设置或返回 input image 的 name 属性值。
src	设置或返回 input image 的 src 属性值。
type	返回 input image 字段的表单元素类型。
value	设置或返回 input image 的 value 属性值。
width	设置或返回 input image 的 width 属性值。

标准属性和事件

Input Image 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Month 对象

Month 对象

Month 对象是 HTML5 中的新对象。

Month 对象表示 HTML <input type="month"> 元素。

注释：Internet Explorer 或 Firefox 不支持 <input type="month"> 元素。

访问 Month 对象

您可以通过使用 getElementById() 来访问 <input type="month"> 元素：

```
var x = document.getElementById("myMonth");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Month 对象。

创建 Month 对象

您可以通过使用 document.createElement() 方法来创建 <input type="month"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "month");
```

Month 对象属性

属性	描述
autocomplete	设置或返回 month 字段的 autocomplete 属性值。
autofocus	设置或返回 month 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 month 字段的默认值。
disabled	设置或返回 month 字段是否被禁用。
form	返回对包含 month 字段的表单的引用。
list	返回对包含 month 字段的 datalist 的引用。
max	设置或返回 month 字段的 max 属性值。
min	设置或返回 month 字段的 min 属性值。
name	设置或返回 month 字段的 name 属性值。
readOnly	设置或返回 month 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 month 字段。
step	设置或返回 month 字段的 step 属性值。
type	返回 month 字段的表单元素类型。
value	设置或返回 month 字段的 value 属性值。

标准属性和事件

Month 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Number 对象

Number 对象

Number 对象是 HTML5 中的新对象。

Number 对象表示 HTML <input type="number"> 元素。

注释：Internet Explorer 9 以及更早的版本不支持 <input type="number"> 元素。

访问 Number 对象

您可以通过使用 getElementById() 来访问 <number> 元素：

```
var x = document.getElementById("myNumber");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Number 对象。

创建 Number 对象

您可以通过使用 document.createElement() 方法来创建 <number> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "number");
```

Number 对象属性

属性	描述
autocomplete	设置或返回 number 字段的 autocomplete 属性值。
autofocus	设置或返回 number 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 number 字段的默认值。
disabled	设置或返回 number 字段是否被禁用。
form	返回对包含 number 字段的表单的引用。
list	返回对包含 number 字段的 datalist 的引用。
max	设置或返回 number 字段的 max 属性值。
min	设置或返回 number 字段的 min 属性值。
name	设置或返回 number 字段的 name 属性值。
placeholder	设置或返回 number 字段的 placeholder 属性值。
readOnly	设置或返回 number 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 number 字段。
step	设置或返回 number 字段的 step 属性值。
type	返回 number 字段的表单元素类型。
value	设置或返回 number 字段的 value 属性值。

标准属性和事件

Number 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Password 对象

Password 对象

Password 对象代表 HTML 表单中的密码字段。

HTML 的 <input type="password"> 标签在表单上每出现一次，一个 Password 对象就会被创建。

该文本输入字段供用户输入某些敏感的数据，比如密码等。当用户输入的时候，他的输入是被掩盖的（例如使用星号*），以防止旁边的人从他背后看到输入的内容。不过需要注意的是，当表单提交时，输入是用明文发送的。

与类型为 "text" 的元素类似，当用户改变显示值时，它会触发 onchange 事件句柄。

您可以通过遍历表单的 elements[] array 来访问密码字段，或者通过使用 document.getElementById() 。

Password 对象属性

属性	描述
accessKey	设置或返回访问密码字段的快捷键。
alt	设置或返回当不支持密码字段时显示的替代文字。
defaultValue	设置或返回密码字段的默认值。
disabled	设置或返回是否应被禁用密码字段。
form	返回对包含此密码字段的表单的引用。
id	设置或返回密码字段的 id。
maxLength	设置或返回密码字段中字符的最大数目。
name	设置或返回密码字段的名称。
readOnly	设置或返回密码字段是否应当是只读的。
size	设置或返回密码字段的长度。
tabIndex	设置或返回密码字段的 tab 键控制次序。
type	返回密码字段的表单元素类型。
value	设置或返回密码字段的 value 属性的值。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Password 对象方法

属性	描述
blur()	从密码字段移开焦点。
focus()	为密码字段赋予焦点。
select()	选取密码字段中的文本。

HTML DOM Input Range 对象

Input Range 对象

Input Range 对象是 HTML5 中的新对象。

Input Range 对象表示 HTML <input type="range"> 元素。

注释：Internet Explorer 9 以及更早的版本不支持 <input type="range"> 元素。

访问 Input Range 对象

您可以通过使用 getElementById() 来访问 <input type="range"> 元素：

```
var x = document.getElementById("myRange");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Input Range 对象。

创建 Input Range 对象

您可以通过使用 document.createElement() 方法来创建 <input type="range"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "range");
```

Input Range 对象属性

属性	描述
autocomplete	设置或返回滑块控件的 autocomplete 属性值。
autofocus	设置或返回滑块控件在页面加载后是否应自动获取焦点。
defaultValue	设置或返回滑块控件的默认值。
disabled	设置或返回滑块控件是否被禁用。
form	返回对包含滑块控件的表单的引用。
list	返回对包含滑块控件的 datalist 的引用。
max	设置或返回滑块控件的 max 属性值。
min	设置或返回滑块控件的 min 属性值。
name	设置或返回滑块控件的 name 属性值。
step	设置或返回滑块控件的 step 属性值。
type	返回滑块控件的表单元素类型。
value	设置或返回滑块控件的 value 属性值。

标准属性和事件

Input Range 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Radio 对象

Radio 对象

Radio 对象代表 HTML 表单中的单选按钮。

在 HTML 表单中 <input type="radio"> 每出现一次，一个 Radio 对象就会被创建。

单选按钮是表示一组互斥选项按钮中的一个。当一个按钮被选中，之前选中的按钮就变为非选中的。

当单选按钮被选中或不选中时，该按钮就会触发 onclick 事件句柄。

您可通过遍历表单的 elements[] 数组来访问 Radio 对象，或者通过使用 document.getElementById()。

Radio 对象属性

属性	描述
accessKey	设置或返回访问单选按钮的快捷键。
alt	设置或返回在不支持单选按钮时显示的替代文本。
checked	设置或返回单选按钮的状态。
defaultChecked	返回单选按钮的默认状态。
disabled	设置或返回是否禁用单选按钮。
form	返回一个对包含此单选按钮的表单的引用。
id	设置或返回单选按钮的 id。
name	设置或返回单选按钮的名称。
tabIndex	设置或返回单选按钮的 tab 键控制次序。
type	返回单选按钮的表单类型。
value	设置或返回单选按钮的 value 属性的值。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Radio 对象方法

方法	描述
blur()	从单选按钮移开焦点。
click()	在单选按钮上模拟一次鼠标点击。
focus()	为单选按钮赋予焦点。

HTML DOM Reset 对象

Reset 对象

Reset 对象代表 HTML 表单中的一个重置按钮。

在 HTML 表单中 <input type="reset"> 标签每出现一次，一个 Reset 对象就会被创建。

当重置按钮被点击，包含它的表单中所有输入元素的值都重置为它们的默认值。默认值由 HTML value 属性或 JavaScript 的 defaultValue 属性指定。

重置按钮在重置表单之前触发 onclick 句柄，并且这个句柄可以通过返回 fasle 来取消。

参阅 Form.reset() 方法 和 Form.onreset 事件句柄。

您可以通过遍历表单的 elements[] 数组来访问某个重置按钮，或者通过使用 document.getElementById()。

Reset 对象属性

属性	描述
accesskey	设置或返回访问重置按钮的快捷键。
alt	设置或返回当浏览器不支持重置按钮时供显示的替代文本。
disabled	设置或返回重置按钮是否应被禁用。
form	返回一个对包含此重置按钮的表单对象的引用。
id	设置或返回重置按钮的 id。
name	设置或返回重置按钮的名称。
tabIndex	设置或返回重置按钮的 tab 键控制次序。
type	返回重置按钮的表单元素类型。
value	设置或返回重置按钮上显示的文本。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Reset 对象方法

方法	描述
blur()	从重置按钮上移开焦点。
click()	在重置按钮上模拟一次鼠标点击。
focus()	为重置按钮赋予焦点。

HTML DOM Input Search 对象

Input Search 对象

Input Search 对象是 HTML5 中的新对象。

Input Search 对象表示 HTML <input type="search"> 元素。

注释：Internet Explorer 9（以及更早的版本）或 Safari 不支持 <input type="search"> 元素。

访问 Input Search 对象

您可以通过使用 getElementById() 来访问 <search> 元素：

```
var x = document.getElementById("mySearch");
```

创建 Input Search 对象

您可以通过使用 document.createElement() 方法来创建 <search> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "search");
```

Input Search 对象属性

属性	描述
autocomplete	设置或返回 Search 字段的 autocomplete 属性值。
autofocus	设置或返回 Search 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 Search 字段的默认值。
disabled	设置或返回 Search 字段是否被禁用。
form	返回对包含 Search 字段的表单的引用。
list	返回对包含 Search 字段的 datalist 的引用。
maxLength	设置或返回 search 字段的 maxLength 属性值。
name	设置或返回 Search 字段的 name 属性值。
pattern	设置或返回 Search 字段的 pattern 属性值。
placeholder	设置或返回 Search 字段的 placeholder 属性值。
readOnly	设置或返回 Search 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 Search 字段。
size	设置或返回 Search 字段的 size 属性值。
step	设置或返回 Search 字段的 step 属性值。
type	返回 Search 字段的表单元素类型。
value	设置或返回 Search 字段的 value 属性值。

标准属性和事件

Input Search 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Submit 对象

Submit 对象

Submit 对象代表 HTML 表单中的一个提交按钮 (submit button)。

在 HTML 表单中 <input type="submit"> 标签每出现一次，一个 Submit 对象就会被创建。

在表单提交之前，触发 onclick 事件句柄，并且一个句柄可以通过返回 fasle 来取消表单提交。

参阅 Form.submit() 方法 和 Form.onsubmit 事件句柄。

实例：表单验证

您可以通过遍历表单的 elements[] 数组来访问某个提交按钮，或者通过使用 document.getElementById()。

Submit 对象属性

属性	描述
accessKey	设置或返回访问提交按钮的快捷键。
alt	设置或返回当浏览器不支持提交按钮时供显示的替代文本。
disabled	设置或返回提交按钮是否应被禁用。
form	返回一个对包含此提交按钮的表单的引用。
id	设置或返回提交按钮的 id。
name	设置或返回提交按钮的名称。
tabIndex	设置或返回提交按钮的 tab 键控制次序。
type	返回提交按钮的表单元素类型。
value	设置或返回在提交按钮上显示的文本。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Submit 对象方法

方法	描述
blur()	从提交按钮上移开焦点。
click()	在提交按钮上模拟一次鼠标点击。
focus()	为提交按钮赋予焦点。

HTML DOM Text 对象

Text 对象

Text 对象代表 HTML 表单中的文本输入域。

在 HTML 表单中 <input type="text"> 每出现一次，Text 对象就会被创建。

该元素可创建一个单行的文本输入字段。当用户编辑显示的文本并随后把输入焦点转移到其他元素的时候，会触发 onchange 事件句柄。

您可以使用 HTML <textarea> 标记来创建多行文本输入。参阅 Textarea 对象。

对于掩码文本输入，把 <input type="text"> 中的 type 设置为 "password"。参阅 Input Password。

您可以通过表单的 elements[] 数组来访问文本输入域，或者通过使用 document.getElementById()。

Text 对象属性

属性	描述
accessKey	设置或返回访问文本域的快捷键。
alt	设置或返回当浏览器不支持文本域时供显示的替代文本。
defaultValue	设置或返回文本域的默认值。
disabled	设置或返回文本域是否应被禁用。
form	返回一个对包含文本域的表单对象的引用。
id	设置或返回文本域的 id。
maxLength	设置或返回文本域中的最大字符数。
name	设置或返回文本域的名称。
readOnly	设置或返回文本域是否应是只读的。
size	设置或返回文本域的尺寸。
tabIndex	设置或返回文本域的 tab 键控制次序。
type	返回文本域的表单元素类型。
value	设置或返回文本域的 value 属性的值。

标准属性

属性	描述
className	设置或返回元素的 class 属性。
dir	设置或返回文本的方向。
lang	设置或返回元素的语言代码。
title	设置或返回元素的 title 属性。

Text 对象方法

方法	描述
blur()	从文本域上移开焦点。
focus()	在文本域上设置焦点。
select()	选取文本域中的内容。

HTML DOM Input Time 对象

Input Time 对象

Input Time 对象是 HTML5 中的新对象。

Input Time 对象表示 HTML <input type="time"> 元素。

注释：Internet Explorer 或 Firefox 不支持 <input type="time"> 元素。

访问 Input Time 对象

您可以通过使用 getElementById() 来访问 HTML <input type="time"> 元素：

```
var x = document.getElementById("myTime");
```

创建 Input Time 对象

您可以通过使用 document.createElement() 方法来创建HTML <input type="time"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "time");
```

Input Time 对象属性

属性	描述
autocomplete	设置或返回 time 字段的 autocomplete 属性值。
autofocus	设置或返回 time 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 time 字段的默认值。
disabled	设置或返回 time 字段是否被禁用。
form	返回对包含 time 字段的表单的引用。
list	返回对包含 time 字段的 datalist 的引用。
max	设置或返回 time 字段的 max 属性值。
min	设置或返回 time 字段的 min 属性值。
name	设置或返回 time 字段的 name 属性值。
readOnly	设置或返回 time 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 time 字段。
size	设置或返回 time 字段的 size 属性值。
step	设置或返回 time 字段的 step 属性值。
type	返回 time 字段的表单元素类型。
value	设置或返回 time 字段的 value 属性值。

标准属性和事件

Input Time 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Input URL 对象

Input URL 对象

Input URL 对象是 HTML5 中的新对象。

Input URL 对象表示 HTML <input type="url"> 元素。

注释：Internet Explorer 9（以及更早的版本）或 Safari 不支持 <input type="search"> 元素。

访问 Input URL 对象

您可以通过使用 getElementById() 来访问 <input type="url"> 元素：

```
var x = document.getElementById("myUrl");
```

创建 Input URL 对象

您可以通过使用 document.createElement() 方法来创建 <input type="url"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "url");
```

Input URL 对象属性

属性	描述
autocomplete	设置或返回 Search 字段的 autocomplete 属性值。
autofocus	设置或返回 Search 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 Search 字段的默认值。
disabled	设置或返回 Search 字段是否被禁用。
form	返回对包含 Search 字段的表单的引用。
list	返回对包含 Search 字段的 datalist 的引用。
maxLength	设置或返回 search 字段的 maxLength 属性值。
multiple	设置或返回是否允许用户在 URL 字段中输入不止一条 url 地址。
name	设置或返回 Search 字段的 name 属性值。
pattern	设置或返回 Search 字段的 pattern 属性值。
placeholder	设置或返回 Search 字段的 placeholder 属性值。
readOnly	设置或返回 Search 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 Search 字段。
size	设置或返回 Search 字段的 size 属性值。
step	设置或返回 Search 字段的 step 属性值。
type	返回 Search 字段的表单元素类型。
value	设置或返回 Search 字段的 value 属性值。

标准属性和事件

Input URL 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML DOM Input Week 对象

Input Week 对象

Input Week 对象是 HTML5 中的新对象。

Input Week 对象表示 HTML <input type="week"> 元素。

注释：Internet Explorer 或 Firefox 不支持 <input type="week"> 元素。

访问 Input Week 对象

您可以通过使用 getElementById() 来访问 <input type="week"> 元素：

```
var x = document.getElementById("myWeek");
```

提示：您也可以通过遍历表单的 elements 集合来访问 Input Week 对象。

创建 Input Week 对象

您可以通过使用 document.createElement() 方法来创建 <input type="week"> 元素：

```
var x = document.createElement("INPUT");
x.setAttribute("type", "week");
```

Input Week 对象属性

属性	描述
autocomplete	设置或返回 week 字段的 autocomplete 属性值。
autofocus	设置或返回 week 字段在页面加载后是否应自动获取焦点。
defaultValue	设置或返回 week 字段的默认值。
disabled	设置或返回 week 字段是否被禁用。
form	返回对包含 week 字段的表单的引用。
list	返回对包含 week 字段的 datalist 的引用。
max	设置或返回 week 字段的 max 属性值。
min	设置或返回 week 字段的 min 属性值。
name	设置或返回 week 字段的 name 属性值。
readOnly	设置或返回 week 字段是否是只读的。
required	设置或返回在提交表单之前是否必须填写 week 字段。
step	设置或返回 week 字段的 step 属性值。
type	返回 week 字段的表单元素类型。
value	设置或返回 week 字段的 value 属性值。

标准属性和事件

Input Week 对象支持标准属性和事件。

相关页面

HTML 教程：HTML 表单

HTML 参考手册：HTML <input> 标签

HTML 参考手册：HTML <input> type 属性

HTML <ins> 标签

实例

带有已删除部分和新插入部分的文本：

```
a dozen is <del>20</del> <ins>12</ins> pieces
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <ins> 标签。

注释：没有主流浏览器能够正确地显示 <ins> 标签的 cite 或 datetime 属性。

定义和用法

<ins> 标签定义已经被插入文档中的文本。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

提示：请与 <del> 一同使用，来描述文档中的更新和修正。

可选的属性

属性	值	描述
cite	URL	指向另外一个文档的 URL，此文档可解释文本被插入的原因。
datetime	YYYYMMDD	定义文本被插入的日期和时间。

全局属性

<ins> 标签支持 HTML 中的全局属性。

事件属性

<ins> 标签支持 HTML 中的事件属性。

TIY 实例

删除字效果和插入字效果

此例演示如何标示删除文本和插入文本。

```
<html>

<body>

<p>一打有 <del>二十</del> <ins>十二</ins> 件。</p>

<p>大多数浏览器会改写为删除文本和下划线文本。</p>

<p>一些老式的浏览器会把删除文本和下划线文本显示为普通文本。</p>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Ins 对象

HTML <keygen> 标签

实例

带有 keygen 字段的表单：

```
<form action="demo_keygen.asp" method="get">
Username: <input type="text" name="usr_name" />
Encryption: <keygen name="security" />
<input type="submit" />
</form>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Firefox, Chrome, Opera 以及 Safari 6 支持 <keygen> 标签。

定义和用法

<keygen> 标签规定用于表单的密钥对生成器字段。

当提交表单时，私钥存储在本地，公钥发送到服务器。

浏览器支持

所有主流浏览器都支持 <keygen> 标签，除了 Internet Explorer 和 Safari。

HTML 4.01 与 HTML 5 之间的差异

keygen 是 HTML 中的新元素。

属性

new : HTML5 中的新属性。

属性	值	描述
autofocus	autofocus	使 keygen 字段在页面加载时获得焦点。
challenge	challenge	如果使用，则将 keygen 的值设置为在提交时询问。
disabled	disabled	禁用 keytag 字段。
form	*formname*	定义该 keygen 字段所属的一个或多个表单。
keytype	rsa	定义 keytype。rsa 生成 RSA 密钥。
name	*fieldname*	定义 keygen 元素的唯一名称。name 属性用于在提交表单时搜集字段的值。

全局属性

<keygen> 标签支持 HTML 中的全局属性。

事件属性

<keygen> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Keygen 对象

HTML <label> 标签

实例

带有两个输入字段和相关标记的简单 HTML 表单：

```
<form>
  <label for="male">Male</label>
  <input type="radio" name="sex" id="male" />
  <br />
  <label for="female">Female</label>
  <input type="radio" name="sex" id="female" />
</form>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <label> 标签。

Safari 2 或更早的版本不支持 <label> 标签。

定义和用法

<label> 标签为 input 元素定义标注（标记）。

label 元素不会向用户呈现任何特殊效果。不过，它为鼠标用户改进了可用性。如果您在 label 元素内点击文本，就会触发此控件。就是说，当用户选择该标签时，浏览器就会自动将焦点转到和标签相关的表单控件上。

<label> 标签的 for 属性应当与相关元素的 id 属性相同。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释："for" 属性可把 label 绑定到另外一个元素。请把 "for" 属性的值设置为相关元素的 id 属性的值。

属性

new : HTML5 中的新属性。

属性	值	描述
for	*id*	规定 label 绑定到哪个表单元素。
form	*formid*	规定 label 字段所属的一个或多个表单。

全局属性

<label> 标签支持 HTML 中的全局属性。

事件属性

<label> 标签支持 HTML 中的事件属性。

TIY 实例

label

如何为控件定义标记。

```
<html>
<body>

<p>请点击文本标记之一，就可以触发相关控件：</p>

<form>
<label for="male">Male</label>
<input type="radio" name="sex" id="male" />
<br />
<label for="female">Female</label>
<input type="radio" name="sex" id="female" />
</form>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Column 对象

HTML <legend> 标签

实例

组合表单中的相关元素：

```
<form>
  <fieldset>
    <legend>health information</legend>
    height: <input type="text" />
    weight: <input type="text" />
  </fieldset>
</form>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <legend> 标签。

定义和用法

legend 元素为 fieldset 元素定义标题（caption）。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，align 属性不被赞成使用。

在 XHTML 1.0 Strict DTD 中，align 属性不被支持。

可选的属性

DTD 指示此属性允许在哪种 DTD 中使用。S=Strict, T=Transitional, F=Frameset.

属性	值	描述	DTD
align	`top` `bottom` `left` `right`	不赞成使用。请使用样式代替。为 fieldset 中的标题定义对齐方式。	TF

全局属性

<fieldset> 标签支持 HTML 中的全局属性。

事件属性

<fieldset> 标签支持 HTML 中的事件属性。

TIY 实例

围绕数据的 Fieldset

本例演示如何在数据周围绘制一个带标题的框。

```
<!DOCTYPE HTML>
<html>

<body>

<form>
  <fieldset>
    <legend>健康信息</legend>
    身高：<input type="text" />
    体重：<input type="text" />
  </fieldset>
</form>

<p>如果表单周围没有边框，说明您的浏览器太老了。</p>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Legend 对象

HTML <li> 标签

实例

```
<ol>
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ol>

<ul>
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ul>
```

（请在页面底部查看更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <li> 标签。

定义和用法

 标签定义列表项目。

 标签可用在有序列表 (<ol>) 和无序列表 (<ul>) 中。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，li 元素的 "type" 和 "value" 属性是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，li 元素的 "type" 和 "value" 属性是不被支持的。

提示和注释：

提示：请使用 CSS 来定义列表和列表项目的类型。

可选的属性

属性	值	描述
type	`A` `a` `I` `i` `1` `disc` `square` `circle`	不赞成使用。请使用样式取代它。规定使用哪种项目符号。
value	*number*	不赞成使用。请使用样式取代它。规定列表项目的数字。

全局属性

 标签支持 HTML 中的全局属性。

事件属性

 标签支持 HTML 中的事件属性。

TIY 实例

一个无序列表

本例演示一个无序列表。

```
<html>

<body>

<h4>一个无序列表：</h4>
<ul>
  <li>咖啡</li>
  <li>茶</li>
  <li>牛奶</li>
</ul>

</body>
</html>
```

不同类型的无序列表

本例演示不同类型的无序列表。

```
<html>
<body>

<h4>Disc 项目符号列表：</h4>
<ul type="disc">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ul>

<h4>Circle 项目符号列表：</h4>
<ul type="circle">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ul>

<h4>Square 项目符号列表：</h4>
<ul type="square">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ul>

</body>
</html>
```

一个有序列表

本例演示一个有序列表。

```
<!DOCTYPE html>
<html>
<body>

<ol>
  <li>咖啡</li>
  <li>牛奶</li>
  <li>茶</li>
</ol>

<ol start="50">
  <li>咖啡</li>
  <li>牛奶</li>
  <li>茶</li>
</ol>

</body>
</html>
```

不同类型的有序列表

本例演示不同类型的有序列表。

```
<html>
<body>

<h4>数字列表：</h4>
<ol>
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ol>

<h4>字母列表：</h4>
<ol type="A">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ol>

<h4>小写字母列表：</h4>
<ol type="a">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ol>

<h4>罗马字母列表：</h4>
<ol type="I">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ol>

<h4>小写罗马字母列表：</h4>
<ol type="i">
 <li>苹果</li>
 <li>香蕉</li>
 <li>柠檬</li>
 <li>桔子</li>
</ol>

</body>
</html>
```

嵌套列表

本例演示如何嵌套列表。

```
<html>

<body>

<h4>一个嵌套列表：</h4>
<ul>
  <li>咖啡</li>
  <li>茶
    <ul>
    <li>红茶</li>
    <li>绿茶</li>
    </ul>
  </li>
  <li>牛奶</li>
</ul>

</body>
</html>
```

嵌套列表 2

本例演示更复杂的嵌套列表。

```
<html>

<body>

<h4>一个嵌套列表：</h4>
<ul>
  <li>咖啡</li>
  <li>茶
    <ul>
    <li>红茶</li>
    <li>绿茶
      <ul>
      <li>中国茶</li>
      <li>非洲茶</li>
      </ul>
    </li>
    </ul>
  </li>
  <li>牛奶</li>
</ul>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Li 对象

HTML <link> 标签

实例

链接一个外部样式表：

```
<head>
<link rel="stylesheet" type="text/css" href="theme.css" />
</head>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

在用于样式表时，<link> 标签得到了几乎所有浏览器的支持。但是几乎没有浏览器支持其他方面的用途。

定义和用法

<link> 标签定义文档与外部资源的关系。

<link> 标签最常见的用途是链接样式表。

HTML 与 XHTML 之间的差异

在 HTML 中，<link> 标签没有结束标签。

在 XHTML 中，<link> 标签必须被正确地关闭。

提示和注释：

注释：link 元素是空元素，它仅包含属性。

注释：此元素只能存在于 head 部分，不过它可出现任何次数。

属性

new : HTML5 中的新属性。

属性	值	描述
charset	*char_encoding*	HTML5 中不支持。
href	*URL*	规定被链接文档的位置。
hreflang	*language_code*	规定被链接文档中文本的语言。
media	*media_query*	规定被链接文档将被显示在什么设备上。
rel	`alternate` `author` `help` `icon` `licence` `next` `pingback` `prefetch` `prev` `search` `sidebar` `stylesheet` `tag`	规定当前文档与被链接文档之间的关系。
rev	*reversed relationship*	HTML5 中不支持。
sizes	*heightxwidth* any	规定被链接资源的尺寸。仅适用于 rel="icon"。
target	`_blank` `_self` `_top` `_parent` `frame_name`	HTML5 中不支持。
type	*MIME_type*	规定被链接文档的 MIME 类型。

全局属性

<link> 标签支持 HTML 中的全局属性。

事件属性

<link> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Link 对象

HTML <main> 标签

实例

```
<main>
  <h1>Web Browsers</h1>
  <p>Google Chrome、Firefox 以及 Internet Explorer 是目前最流行的浏览器

  <article>
    <h1>Google Chrome</h1>
    <p>Google Chrome 是由 Google 开发的一款免费的开源 web 浏览器，于 200
  </article>

  <article>
    <h1>Internet Explorer</h1>
    <p>Internet Explorer 由微软开发的一款免费的 web 浏览器，发布于 1995 
  </article>

  <article>
    <h1>Mozilla Firefox</h1>
    <p>Firefox 是一款来自 Mozilla 的免费开源 web 浏览器，发布于 2004 年。
  </article>
</main>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <main> 标签，除了 Internet Explorer。

定义和用法

<main> 标签规定文档的主要内容。

<main> 元素中的内容对于文档来说应当是唯一的。它不应包含在文档中重复出现的内容，比如侧栏、导航栏、版权信息、站点标志或搜索表单。

注释：在一个文档中，不能出现一个以上的 <main> 元素。<main> 元素不能是以下元素的后代：<article>、<aside>、<footer>、<header> 或 <nav>。

HTML 4.01 与 HTML 5 之间的差异

<main> 标签是 HTML 5 中的新标签。

全局属性

<main> 标签支持 HTML 中的全局属性。

事件属性

<main> 标签支持 HTML 中的事件属性。

HTML <map> 标签

实例

带有可点击区域的图像映射：

```
<img src="planets.jpg" border="0" usemap="#planetmap" alt="Planets'

<map name="planetmap" id="planetmap">
  <area shape="circle" coords="180,139,14" href ="venus.html" alt='
  <area shape="circle" coords="129,161,10" href ="mercur.html" alt=
  <area shape="rect" coords="0,0,110,260" href ="sun.html" alt="Sur
</map>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <map> 标签。

定义和用法

定义一个客户端图像映射。图像映射（image-map）指带有可点击区域的一幅图像。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：area 元素永远嵌套在 map 元素内部。area 元素可定义图像映射中的区域。

注释：<img>中的 usemap 属性可引用 <map> 中的 id 或 name 属性（取决于浏览器），所以我们应同时向 <map> 添加 id 和 name 属性。

必需的属性

属性	值	描述
id	unique_name	为 map 标签定义唯一的名称。

可选的属性

属性	值	描述
name	*mapname*	为 image-map 规定的名称。

全局属性

<map> 标签支持 HTML 中的全局属性。

事件属性

<map> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Map 对象

HTML <mark> 标签

实例

突出显示部分文本：

```
<p>Do not forget to buy <mark>milk</mark> today.</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <mark> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <mark> 标签。

定义和用法

<mark> 标签定义带有记号的文本。请在需要突出显示文本时使用 <m> 标签。

HTML 4.01 与 HTML 5 之间的差异

<mark> 标签是 HTML 5 中的新标签。

全局属性

<mark> 标签支持 HTML 中的全局属性。

事件属性

<mark> 标签支持 HTML 中的事件属性。

HTML <menu> 标签

实例

带有两个菜单按钮（"File" 和 "Edit"）的工具栏，每个按钮都包含带有一系列选项的下拉列表：

```
<menu type="toolbar">
 <li>
 <menu label="File">
 <button type="button" onclick="file_new()">New...</button>
 <button type="button" onclick="file_open()">Open...</button>
 <button type="button" onclick="file_save()">Save</button>
 </menu>
 </li>
 <li>
 <menu label="Edit">
 <button type="button" onclick="edit_cut()">Cut</button>
 <button type="button" onclick="edit_copy()">Copy</button>
 <button type="button" onclick="edit_paste()">Paste</button>
 </menu>
 </li>
</menu>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前所有主流浏览器都不支持 <menu> 标签。

定义和用法

<menu> 标签定义命令的列表或菜单。

<menu> 标签用于上下文菜单、工具栏以及用于列出表单控件和命令。

提示和注释：

提示：请使用 CSS 来设置菜单列表的样式！

HTML 4.01 与 HTML5 之间的差异

在 HTML 4.01 中已弃用 <menu> 元素。

在 HTML5 中重新定义了 <menu> 元素。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，menu 元素不被赞成使用。

在 XHTML 1.0 Strict DTD 中，menu 元素不被支持。

属性

New: HTML5 中的新属性。

属性	值	描述
label	text	规定菜单的可见标签。
type	`popup` `toolbar`	规定要显示哪种菜单类型。

全局属性

<menu> 标签支持 HTML 中的全局属性。

事件属性

<menu> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Column 对象

HTML <menuitem> 标签

实例

包含不同 <menuitem> 元素的上下文菜单：

```
<menu type="context" id="mymenu">
  <menuitem label="Refresh" onclick="window.location.reload();" icc
  </menuitem>
  <menu label="Share on...">
    <menuitem label="Twitter" icon="ico_twitter.png"
    onclick="window.open('//twitter.com/intent/tweet?text='+window.
    </menuitem>
    <menuitem label="Facebook" icon="ico_facebook.png"
    onclick="window.open('//facebook.com/sharer/sharer.php?u='+winc
    </menuitem>
  </menu>
  <menuitem label="Email This Page"
  onclick="window.location='mailto:?body='+window.location.href;"><
</menu>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Firefox 8.0 以及更高的版本支持 <menuitem> 标签。

定义和用法

<menuitem> 标签定义用户可以从弹出菜单调用的命令/菜单项目。

HTML 4.01 与 HTML 5 之间的差异

<menuitem> 标签是 HTML5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
checked	checked	规定在页面加载后选中命令/菜单项目。仅适用于 type="radio" 或 type="checkbox"。
default	default	把命令/菜单项设置为默认命令。
disabled	disabled	规定命令/菜单项应该被禁用。
icon	URL	规定命令/菜单项的图标。
open	open	定义 details 是否可见。
label	*text*	必需。规定命令/菜单项的名称，以向用户显示。
radiogroup	*groupname*	规定命令组的名称，命令组会在命令/菜单项本身被切换时进行切换。仅适用于 type="radio"。
type	`checkbox` `command` `radio`	规定命令/菜单项的类型。默认是 "command"。

全局属性

<menuitem> 标签支持 HTML 中的全局属性。

事件属性

<menuitem> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：MenuItem 对象

HTML <meta> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <meta> 标签。

定义和用法

<meta> 元素可提供有关页面的元信息（meta-information），比如针对搜索引擎和更新频度的描述和关键词。

<meta> 标签位于文档的头部，不包含任何内容。<meta> 标签的属性定义了与文档相关联的名称/值对。

HTML 与 XHTML 之间的差异

在 HTML 中，<meta> 标签没有结束标签。

在 XHTML 中，<meta> 标签必须被正确地关闭。

提示和注释：

注释：<meta> 标签永远位于 head 元素内部。

注释：元数据总是以名称/值的形式被成对传递的。

必需的属性

属性	值	描述
content	some_text	定义与 http-equiv 或 name 属性相关的元信息

可选的属性

属性	值	描述
http-equiv	`content-type` `expires` `refresh` `set-cookie`	把 content 属性关联到 HTTP 头部。
name	`author` `description` `keywords` `generator` `revised` `others`	把 content 属性关联到一个名称。
scheme	some_text	定义用于翻译 content 属性值的格式。

name 属性

name 属性提供了名称/值对中的名称。HTML 和 XHTML 标签都没有指定任何预先定义的 <meta> 名称。通常情况下，您可以自由使用对自己和源文档的读者来说富有意义的名称。

"keywords" 是一个经常被用到的名称。它为文档定义了一组关键字。某些搜索引擎在遇到这些关键字时，会用这些关键字对文档进行分类。

类似这样的 meta 标签可能对于进入搜索引擎的索引有帮助：

```
<meta name="keywords" content="HTML,ASP,PHP,SQL">
```

如果没有提供 name 属性，那么名称/值对中的名称会采用 http-equiv 属性的值。

http-equiv 属性

http-equiv 属性为名称/值对提供了名称。并指示服务器在发送实际的文档之前先在要传送给浏览器的 MIME 文档头部包含名称/值对。

当服务器向浏览器发送文档时，会先发送许多名称/值对。虽然有些服务器会发送许多这种名称/值对，但是所有服务器都至少要发送一个：content-type:text/html。这将告诉浏览器准备接受一个 HTML 文档。

使用带有 http-equiv 属性的 <meta> 标签时，服务器将把名称/值对添加到发送给浏览器的内容头部。例如，添加：

```
<meta http-equiv="charset" content="iso-8859-1">
<meta http-equiv="expires" content="31 Dec 2008">
```

这样发送到浏览器的头部就应该包含：

```
content-type: text/html
charset:iso-8859-1
expires:31 Dec 2008
```

当然，只有浏览器可以接受这些附加的头部字段，并能以适当的方式使用它们时，这些字段才有意义。

content 属性

content 属性提供了名称/值对中的值。该值可以是任何有效的字符串。

content 属性始终要和 name 属性或 http-equiv 属性一起使用。

scheme 属性

scheme 属性用于指定要用来翻译属性值的方案。此方案应该在由 <head> 标签的 profile 属性指定的概况文件中进行了定义。

全局属性

<meta> 标签支持 HTML 中的全局属性。

相关页面

HTML DOM 参考手册：Meta 对象

HTML <meter> 标签

实例

使用 meter 元素来度量给定范围（gauge）内的数据：

```
<meter value="3" min="0" max="10">十分之三</meter>

<meter value="0.6">60%</meter>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Firefox, Chrome, Opera 以及 Safari 6 支持 <meter> 标签。

定义和用法

<meter> 标签定义已知范围或分数值内的标量测量。也被称为 gauge（尺度）。

例子：磁盘用量、查询结果的相关性，等等。

注释：<meter> 标签不应用于指示进度（在进度条中）。如果标记进度条，请使用 <progress> 标签。

HTML 4.01 与 HTML 5 之间的差异

<meter> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
form	*form_id*	规定 <meter> 元素所属的一个或多个表单。
high	*number*	规定被视作高的值的范围。
low	*number*	规定被视作低的值的范围。
max	*number*	规定范围的最大值。
min	*number*	规定范围的最小值。
optimum	*number*	规定度量的优化值。
value	*number*	必需。规定度量的当前值。

全局属性

<meter> 标签支持 HTML 中的全局属性。

事件属性

<meter> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Meter 对象

相关页面

HTML DOM 参考手册：Meter 对象

HTML <nav> 标签

实例

```
<nav>
<a href="index.asp">Home</a>
<a href="html5_meter.asp">Previous</a>
<a href="html5_noscript.asp">Next</a>
</nav>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9, Firefox, Opera, Chrome 以及 Safari 支持 <nav> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <nav> 标签。

定义和用法

<nav> 标签定义导航链接的部分。

HTML 4.01 与 HTML 5 之间的差异

<nav> 标签是 HTML 5 中的新标签。

提示和注释

提示：如果文档中有“前后”按钮，则应该把它放到 <nav> 元素中。

全局属性

<nav> 标签支持 HTML 中的全局属性。

事件属性

<nav> 标签支持 HTML 中的事件属性。

HTML <noframes> 标签

实例

```
<frameset cols = "25%, 25%,*">
  <noframes>
  <body>Your browser does not handle frames!</body>
  </noframes>
  <frame src ="venus.htm" />
  <frame src  ="sun.htm" />
  <frame src ="mercur.htm"   />
</frameset>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <noframes> 标签。

定义和用法

noframes 元素可为那些不支持框架的浏览器显示文本。noframes 元素位于 frameset 元素内部。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：如果浏览器有能力处理框架，就不会显示出 frameset 元素中的文本。

重要事项：如果您希望 frameset 添加 <noframes> 标签，就必须把其中的文本包装在 <body></body> 标签中！

注释：如果您希望验证包含框架的页面，请确保 DTD 被设置为 "Frameset DTD"。

阅读更多有关 XHTML 验证的内容。

注释：在XHTML 1.0 Strict DTD 中，<noframes> 标签是不被允许的。

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

TIY 实例

如何使用 <noframes> 标签

本例演示：如何使用 <noframes> 标签。

```
<html>

<frameset cols="25%,50%,25%">
  <frame src="/example/html/frame_a.html">
  <frame src="/example/html/frame_b.html">
  <frame src="/example/html/frame_c.html">

<noframes>
<body>您的浏览器无法处理框架！</body>
</noframes>

</frameset>

</html>
```

HTML <noscript> 标签

例子

JavaScript

```
<body>
  ...
  ...

  <script type="text/javascript">
    <!--
    document.write("Hello World!")
    //-->
  </script>

  <noscript>Your browser does not support JavaScript!</noscript>
  ...
  ...
</body>
```

VBScript

```
<body>
  ...
  ...
  <script type="text/vbscript">
   <!--
   document.write("Hello World!")
   '-->
  </script>

  <noscript>Your browser does not support VBScript!</noscript>
  ...
  ...
</body>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <noscript> 标签。

定义和用法

noscript 元素用来定义在脚本未被执行时的替代内容（文本）。

此标签可被用于可识别 <script> 标签但无法支持其中的脚本的浏览器。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：如果浏览器支持脚本，那么它不会显示出 noscript 元素中的文本。

注释：无法识别 <script> 标签的浏览器会把标签的内容显示到页面上。为了避免浏览器这样做，您应当在注释标签中隐藏脚本。老式的（无法识别 <script> 标签的）浏览器会忽略注释，这样就不会把标签的内容写到页面上，而新式的浏览器则懂得执行这些脚本，即使它们被包围在注释标签中！

全局属性

<noscript> 标签支持 HTML 中的全局属性。

HTML <object> 标签

实例

向 HTML 代码添加一个对象：

```
<object classid="clsid:F08DF954-8592-11D1-B16A-00C0F0283628" id="Sl
width="100" height="50">
  <param name="BorderStyle" value="1" />
  <param name="MousePointer" value="0" />
  <param name="Enabled" value="1" />
  <param name="Min" value="0" />
  <param name="Max" value="10" />
</object>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

几乎所有主流浏览器都拥有部分对 <object> 标签的支持。

定义和用法

定义一个嵌入的对象。请使用此元素向您的 XHTML 页面添加多媒体。此元素允许您规定插入 HTML 文档中的对象的数据和参数，以及可用来显示和操作数据的代码。

<object> 标签用于包含对象，比如图像、音频、视频、Java applets、ActiveX、PDF 以及 Flash。

object 的初衷是取代 img 和 applet 元素。不过由于漏洞以及缺乏浏览器支持，这一点并未实现。

浏览器的对象支持有赖于对象类型。不幸的是，主流浏览器都使用不同的代码来加载相同的对象类型。

而幸运的是，object 对象提供了解决方案。如果未显示 object 元素，就会执行位于 <object> 和 </object> 之间的代码。通过这种方式，我们能够嵌套多个 object 元素（每个对应一个浏览器）。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：<param> 标签定义用于对象的 run-time 设置。

注释：不要对图像使用 <object> 标签，请使用 <img> 标签代替。

可选的属性

属性	值	描述

align	`left`
	`right`
	`top`
	`bottom`
		定义围绕该对象的文本对齐方式。

archive	*URL*	由空格分隔的指向档案文件的 URL 列表。这些档案文件包含了与对象相关的资源。
border	*pixels*	定义对象周围的边框。
classid	*class ID*	定义嵌入 Windows Registry 中或某个 URL 中的类的 ID 值，此属性可用来指定浏览器中包含的对象的位置，通常是一个 Java 类。
codebase	*URL*	定义在何处可找到对象所需的代码，提供一个基准 URL。
codetype	*MIME type*	通过 classid 属性所引用的代码的 MIME 类型。
data	*URL*	定义引用对象数据的 URL。如果有需要对象处理的数据文件,要用 data 属性来指定这些数据文件。
declare	declare	可定义此对象仅可被声明，但不能被创建或例示，直到此对象得到应用为止。
form	*form_id*	规定对象所属的一个或多个表单。
height	*pixels*	定义对象的高度。
hspace	*pixels*	定义对象周围水平方向的空白。
name	*unique_name*	为对象定义唯一的名称（以便在脚本中使用）。
standby	*text*	定义当对象正在加载时所显示的文本。
type	*MIME_type*	定义被规定在 data 属性中指定的文件中出现的数据的 MIME 类型。
usemap	*URL*	规定与对象一同使用的客户端图像映射的 URL。
vspace	*pixels*	定义对象的垂直方向的空白。
width	*pixels*	定义对象的宽度。

全局属性

<object> 标签支持 HTML 中的全局属性。

事件属性

<object> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Object 对象

HTML <ol> 标签

实例

有序 HTML 列表：

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <ol> 标签。

定义和用法

 标签定义有序列表。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，ol 元素的 "compact"、"start" 以及 "type" 属性是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，ol 元素的 "compact"、"start" 以及 "type" 属性是不被支持的。

提示和注释：

提示：请使用 CSS 来定义列表的类型。

属性

New：HTML5 中的新属性。

属性	值	描述
compact	compact	HTML5 中不支持。HTML 4.01 中不赞成使用。规定列表呈现的效果比正常情况更小巧。
reversed	reversed	规定列表顺序为降序。(9,8,7...)
start	*number*	规定有序列表的起始值。
type	`1` `A` `a` `I` `i`	规定在列表中使用的标记类型。

全局属性

 标签支持 HTML 中的全局属性。

事件属性

 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Ol 对象

HTML <optgroup> 标签

实例

通过 <optgroup> 标签把相关的选项组合在一起：

```
<select>
  <optgroup label="Swedish Cars">
    <option value ="volvo">Volvo</option>
    <option value ="saab">Saab</option>
  </optgroup>

  <optgroup label="German Cars">
    <option value ="mercedes">Mercedes</option>
    <option value ="audi">Audi</option>
  </optgroup>
</select>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <optgroup> 标签。

定义和用法

<optgroup> 标签定义选项组。

optgroup 元素用于组合选项。当您使用一个长的选项列表时，对相关的选项进行组合会使处理更加容易。

HTML 与 XHTML 之间的差异

NONE

必需的属性

属性	值	描述
label	*text*	为选项组规定描述。

可选的属性

属性	值	描述
disabled	disabled	规定禁用该选项组。

全局属性

<optgroup> 标签支持 HTML 中的全局属性。

事件属性

<optgroup> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：OptionGroup 对象

HTML <option> 标签

实例

创建带有 4 个选项的选择列表：

```
<select>
  <option value ="volvo">Volvo</option>
  <option value ="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <option> 标签。

定义和用法

option 元素定义下拉列表中的一个选项（一个条目）。

浏览器将 <option> 标签中的内容作为 <select> 标签的菜单或是滚动列表中的一个元素显示。

option 元素位于 select 元素内部。

HTML 与 XHTML 之间的差异

在 HTML 中，<option> 没有结束标签。

在 XHTML 中，<option> 必须被正确关闭。

提示和注释：

注释：<option> 标签可以在不带有任何属性的情况下使用，但是您通常需要使用 value 属性，此属性会指示出被送往服务器的内容。

注释：请与 select 元素配合使用此标签，否则这个标签是没有意义的。

提示：如果列表选项很多，可以使用 <optgroup> 标签对相关选项进行组合。

可选的属性

属性	值	描述
disabled	disabled	规定此选项应在首次加载时被禁用。
label	*text*	定义当使用 <optgroup> 时所使用的标注。
selected	selected	规定选项（在首次显示在列表中时）表现为选中状态。
value	*text*	定义送往服务器的选项值。

全局属性

<option> 标签支持 HTML 中的全局属性。

事件属性

<option> 标签支持 HTML 中的事件属性。

TIY 实例

简单的下拉列表

本例演示如何在 HTML 页面中创建简单的下拉列表框。下拉列表框是一个可选列表。

```
<html>

<body>

<form>
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
</form>

</body>
</html>
```

另一个下拉列表

本例演示如何创建一个简单的带有预选值的下拉列表。（注：预选值指预先指定的首选项。）

```
<html>

<body>

<form>
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat" selected="selected">Fiat</option>
<option value="audi">Audi</option>
</select>
</form>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Option 对象

HTML <output> 标签

实例

执行计算然后在 <output> 元素中显示结果：

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
   <input type="range" id="a" value="50">100
   +<input type="number" id="b" value="50">
   =<output name="x" for="a b"></output>
</form>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Firefox, Chrome, Safari 以及 Opera 支持 <output> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <output> 标签。

定义和用法

<output> 标签定义不同类型的输出，比如脚本的输出。

HTML 4.01 与 HTML 5 之间的差异

<output> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
for	*element_id*	定义输出域相关的一个或多个元素。
form	*form_id*	定义输入字段所属的一个或多个表单。
name	*name*	定义对象的唯一名称。（表单提交时使用）

全局属性

<output> 标签支持 HTML 中的全局属性。

事件属性

<output> 标签支持 HTML 中的事件属性。

HTML <p> 标签

实例

以下代码标记了一个段落：

```
<p>This is some text in a very short paragraph</p>
```

（在页面下部，您可以找到更多实例。）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <p> 标签。

定义和用法

<p> 标签定义段落。

p 元素会自动在其前后创建一些空白。浏览器会自动添加这些空间，您也可以在样式表中规定。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，所有 p 元素的呈现属性均不被赞成使用。

在 XHTML 1.0 Strict DTD 中，所有 p 元素的呈现属性均不被支持。

可选的属性

属性	值	描述
align	`left` `right` `center` `justify`	不赞成使用。请使用样式取代它。规定段落中文本的对齐方式。

全局属性

<p> 标签支持 HTML 中的全局属性。

事件属性

<p> 标签支持 HTML 中的事件属性。

TIY 实例

简单的段落

此例演示：段落元素中的文字如何被浏览器显示。

```
<html>
<body>

<p>这是段落。</p>
<p>这是段落。</p>
<p>这是段落。</p>

<p>段落元素由 p 标签定义。</p>

</body>
</html>
```

更多的段落

本例演示段落元素的某些缺省的行为。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<body>

<p>
这个段落
在源代码中
包含许多行
但是浏览器
忽略了它们。
</p>

<p>
这个段落
在源代码          中
包含    许多行
但是          浏览器
忽略了    它们。
</p>

<p>
段落的行数依赖于浏览器窗口的大小。如果调节浏览器窗口的大小，将改变段落中的行数。
</p>

</body>
</html>
```

“诗歌”问题

本例演示 HTML 格式化的某些问题。

```
<html>

<body>

<h1>春晓</h1>

<p>
    春眠不觉晓，
      处处闻啼鸟。
        夜来风雨声，
          花落知多少。
</p>

<p>注意，浏览器忽略了源代码中的排版（省略了多余的空格和换行）。</p>

</body>

</html>
```

延伸阅读：允许的段落用法

可以只在块（block）内指定段落，也可以把段落和其他段落、列表、表单和预定义格式的文本一起使用。总的来讲，这意味着段落可以在任何有合适的文本流的地方出现，例如文档的主体中、列表的元素里，等等。

从技术角度将，段落不可以出现在头部、锚或者其他严格要求内容必须只能是文本的地方。实际上，多数浏览器都忽略了这个限制，它们会把段落作为所含元素的内容一起格式化。

HTML <param> 标签

实例

向 HTML 代码添加一个对象：

```
<object classid="clsid:F08DF954-8592-11D1-B16A-00C0F0283628" id="Sl
width="100" height="50">
  <param name="BorderStyle" value="1" />
  <param name="MousePointer" value="0" />
  <param name="Enabled" value="1" />
  <param name="Min" value="0" />
  <param name="Max" value="10" />
</object>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <param> 标签。

定义和用法

param 元素允许您为插入 XHTML 文档的对象规定 run-time 设置，也就是说，此标签可为包含它的 <object> 或者 <applet> 标签提供参数。

HTML 与 XHTML 之间的差异

在 HTML 中，<param> 标签没有结束标签。

在 XHTML 中，<param> 标签必须被正确地关闭。

必需的属性

属性	值	描述
name	*unique_name*	定义参数的名称（用在脚本中）。

可选的属性

属性	值	描述
type	*MIME type*	规定参数的 MIME 类型（internet media type）。
value	*value*	规定参数的值。
valuetype	`data` `ref` `object`	规定值的 MIME 类型。

全局属性

<param> 标签支持 HTML 中的全局属性。

事件属性

<param> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Parameter 对象

HTML <pre> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <pre> 标签。

定义和用法

pre 元素可定义预格式化的文本。被包围在 pre 元素中的文本通常会保留空格和换行符。而文本也会呈现为等宽字体。

<pre> 标签的一个常见应用就是用来表示计算机的源代码。

可以导致段落断开的标签（例如标题、<p> 和 <address> 标签）绝不能包含在 <pre> 所定义的块里。尽管有些浏览器会把段落结束标签解释为简单地换行，但是这种行为在所有浏览器上并不都是一样的。

pre 元素中允许的文本可以包括物理样式和基于内容的样式变化，还有链接、图像和水平分隔线。当把其他标签（比如 <a> 标签）放到 <pre> 块中时，就像放在 HTML/XHTML 文档的其他部分中一样即可。请看下面的例子：

```
<pre>
&lt;html&gt;

&lt;head&gt;
  &lt;script type=&quot;text/javascript&quot; src=&quot;loadxmldoc
&lt;/script&gt;
&lt;/head&gt;

&lt;body&gt;

  &lt;script type=&quot;text/javascript&quot;&gt;
    xmlDoc=<a href="dom_loadxmldoc.asp">loadXMLDoc</a>(&quot;books
    document.write(&quot;xmlDoc is loaded, ready for use&quot;);
  &lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
</pre>
```

在上面的代码中，<pre> 标签中的特殊符号被转换为符号实体，比如 "<" 代表 "<"，">" 代表 ">"。另外，请注意蓝色的代码，我们在 <pre> 标签中使用了链接，也就是 <a> 标签。上面这段代码的显示效果如下：

```
<html>

<head>
  <script type="text/javascript" src="loadxmldoc.js">
</script>
</head>

<body>

  <script type="text/javascript">
    xmlDoc=[loadXMLDoc](/xmldom/dom_loadxmldoc.asp)("books.xml");
    document.write("xmlDoc is loaded, ready for use");
  </script>

</body>

</html>
```

提示和注释

提示：制表符（tab）在 <pre> 标签定义的块当中可以起到应有的作用，每个制表符占据 8 个字符的位置。但是我们不推荐使用它，因为在不同的浏览器中，Tab 的实现各不相同。在用 <pre> 标签格式化的文档段中使用空格，可以确保文本正确的水平位置。

提示：如果您希望使用 <pre> 标签来定义计算机源代码，比如 HTML 源代码，请使用符号实体来表示特殊字符，比如 "<" 代表 "<"，">" 代表 ">"，"&" 代表 "&"。

提示：在 W3School 中，非常多页面中的源代码实例都是通过 <pre> 标签定义的，您可以参考这些页面，学习如何使用该标签。我们甚至把 <pre> 标签与 <code> 标签结合起来使用，以获得更加精确的语义。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，pre 元素的 "width" 属性是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，pre 元素的 "width" 属性是不被支持的。

可选的属性

属性	值	描述
width	number	定义每行的最大字符数（通常是 40、80 或 132）。

全局属性

<pre> 标签支持 HTML 中的全局属性。

事件属性

<pre> 标签支持 HTML 中的事件属性。

TIY 实例

预格式文本

此例演示如何使用pre标签对空行和空格进行控制。

```
<html>

<body>

<pre>
这是
预格式文本。
它保留了      空格
和换行。
</pre>

<p>pre 标签很适合显示计算机代码：</p>

<pre>
for i = 1 to 10
     print i
next i
</pre>

</body>
</html>
```

HTML <progress> 标签

实例

正在进行的下载：

```
<progress value="22" max="100"></progress>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10, Firefox, Opera, Chrome 以及 Safari 6 支持 <progress> 标签。

注释：Internet Explorer 9 以及更早的版本不支持 <progress> 标签。

定义和用法

<progress> 标签标示任务的进度（进程）。

HTML 4.01 与 HTML 5 之间的差异

<progress> 标签是 HTML 5 中的新标签。

提示和注释

提示：请结合 <progress> 标签与 JavaScript 一同使用，来显示任务的进度。

注释：<progress> 标签不适合用来表示度量衡（例如，磁盘空间使用情况或查询结果）。如需表示度量衡，请使用 <meter> 标签代替。

属性

new : HTML5 中的新属性。

属性	值	描述
max	*number*	规定任务一共需要多少工作。
value	*number*	规定已经完成多少任务。

全局属性

<progress> 标签支持 HTML 中的全局属性。

事件属性

<progress> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Progress 对象

HTML <q> 标签

实例

标记短的引用：

```
<q>Here is a short quotation here is a short quotation</q>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <q> 标签。

定义和用法

<q> 标签定义短的引用。

浏览器经常在引用的内容周围添加引号。

<q> 与 <blockquote> 的区别

<q> 标签在本质上与 <blockquote> 是一样的。不同之处在于它们的显示和应用。<q> 标签用于简短的行内引用。如果需要从周围内容分离出来比较长的部分（通常显示为缩进的块），请使用 <blockquote> 标签。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

提示：根据 HTML 4.01 规范，q 元素应当使用分界引号来呈现，就是说，q 元素包含的文本必须以引号来开始和结束。Mozilla 浏览器 (Firefox) 以及 Opera 符合这个规定，但是 Internet Explorer 却不支持此规定。结果，如果要使用 <q> 标签，而且

用自己的引号来满足 Internet Explorer，那么就要在符合标准的浏览器使用两组引号。尽管如此，我们还是推荐使用 <q> 标签，这不仅仅因为我们喜欢标准，还因为我们预见到了将其应用于文档处理、信息提取等方面的显示效果。

可选的属性

属性	值	描述
cite	citation	定义引用的出处或来源（citation）

全局属性

<abbr> 标签支持 HTML 中的全局属性。

事件属性

<abbr> 标签支持 HTML 中的事件属性。

TIY 实例

引用

此例演示如何实现长短不一的引用语。

```
<html>

<body>

这是长的引用：
<blockquote>
这是长的引用。这是长的引用。这是长的引用。这是长的引用。这是长的引用。这是长的引
</blockquote>

这是短的引用：
<q>
这是短的引用。
</q>

<p>
使用 blockquote 元素的话，浏览器会插入换行和外边距，而 q 元素不会有任何特殊的
</p>

</body>
</html>
```

相关页面

HTML DOM 参考手册：Quote 对象

HTML <rp> 标签

实例

一个 ruby 注释：

```
<ruby>
漢 <rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt>
</ruby>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <rp> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <rp> 标签。

定义和用法

<rp> 标签在 ruby 注释中使用，以定义不支持 ruby 元素的浏览器所显示的内容。

ruby 注释是中文注音或字符。

在东亚使用，显示的是东亚字符的发音。

与 <ruby> 以及 <rt> 标签一同使用：

ruby 元素由一个或多个字符（需要一个解释/发音）和一个提供该信息的 rt 元素组成，还包括可选的 rp 元素，定义当浏览器不支持 "ruby" 元素时显示的内容。

HTML 4.01 与 HTML 5 之间的差异

<rp> 标签是 HTML 5 的新标签。

提示和注释

提示：支持 "ruby" 元素的浏览器不会显示 "rp" 元素的内容。

全局属性

<rp> 标签支持 HTML 中的全局属性。

事件属性

<rp> 标签支持 HTML 中的事件属性。

HTML <rt> 标签

实例

一个 ruby 注释：

```
<ruby>
漢 <rt> ㄏㄢˋ </rt>
</ruby>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <rt> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <rt> 标签。

定义和用法

<rt> 标签定义字符（中文注音或字符）的解释或发音。

ruby 注释是中文注音或字符。

在东亚使用，显示的是东亚字符的发音。

与 <ruby> 以及 <rt> 标签一同使用：

ruby 元素由一个或多个字符（需要一个解释/发音）和一个提供该信息的 rt 元素组成，还包括可选的 rp 元素，定义当浏览器不支持 "ruby" 元素时显示的内容。

HTML 4.01 与 HTML 5 之间的差异

<rt> 标签是 HTML 5 的新标签。

提示和注释

提示：支持 "ruby" 元素的浏览器不会显示 "rp" 元素的内容。

全局属性

<rt> 标签支持 HTML 中的全局属性。

事件属性

<rt> 标签支持 HTML 中的事件属性。

HTML <ruby> 标签

实例

一个 ruby 注释：

```
<ruby>
漢 <rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt>
</ruby>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <ruby> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <ruby> 标签。

定义和用法

<ruby> 标签定义 ruby 注释（中文注音或字符）。

在东亚使用，显示的是东亚字符的发音。

与 <ruby> 以及 <rt> 标签一同使用：

ruby 元素由一个或多个字符（需要一个解释/发音）和一个提供该信息的 rt 元素组成，还包括可选的 rp 元素，定义当浏览器不支持 "ruby" 元素时显示的内容。

HTML 4.01 与 HTML 5 之间的差异

<ruby> 标签是 HTML 5 的新标签。

提示和注释

提示：支持 "ruby" 元素的浏览器不会显示 "rp" 元素的内容。

全局属性

<ruby> 标签支持 HTML 中的全局属性。

事件属性

<ruby> 标签支持 HTML 中的事件属性。

HTML <s> 标签

例子

可以像这样标记删除线文本：

```
在 HTML 5 中，<s>仍然支持</s>已经不支持这个标签了。
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <s> 标签。

定义和用法

<s> 标签可定义加删除线文本定义。

<s> 标签是 <strike> 标签的缩写版本，Netscape 和 Internet Explorer 都支持它。但在 HTML 4 和 XHTML 中已经不再赞成使用它了，意思就是不再使用了；它早晚有一天将会消失。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，<s> 标签不被赞成使用。

在 XHTML 1.0 Strict DTD 中，<s> 标签不被支持。

提示和注释：

注释：请使用 <del> 替代它！

全局属性

<s> 标签支持 HTML 中的全局属性。

事件属性

<s> 标签支持 HTML 中的事件属性。

TIY 实例

删除线文本

此例演示如何为文本标记添加删除线。

```
<html>

<body>

<p>在 HTML 5 中，<s>仍然支持</s>已经不支持这个标签了。</p>

</body>
</html>
```

参阅

请参阅 HTML 5 参考手册，了解有关 <s> 标签的最新信息：

HTML 5 <s> 标签

HTML <script> 标签

实例

在 HTML 页面中插入一段 JavaScript：

```
<script type="text/javascript">
document.write("Hello World!")
</script>
```

（在本页底部可以找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <script> 标签。

定义和用法

<script> 标签用于定义客户端脚本，比如 JavaScript。

script 元素既可以包含脚本语句，也可以通过 src 属性指向外部脚本文件。

必需的 type 属性规定脚本的 MIME 类型。

JavaScript 的常见应用时图像操作、表单验证以及动态内容更新。

提示和注释：

注释：假如此元素内部的代码没有位于某个函数中，那么这些代码会在页面被加载时被立即执行。<frameset> 标签之后的脚本会被忽略。

提示：请参阅 noscript 元素；对于那些在浏览器中禁用脚本或者其浏览器不支持客户端脚本的用户来说，该元素非常有用。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，script 元素的 "language" 属性不被赞成使用。

在 XHTML 1.0 Strict DTD 中，script 元素的 "language" 属性不被支持。

HTML 4 和 XHTML 在处理脚本中的内容方面有所不同：

在 HTML 4 中，内容类型声明为 CDATA，就是说不会对实体进行解析。

在 XHTML 中，内容类型声明为 (#PCDATA)，也就是说会对实体进行解析。

这意味着，在 XHTML 中，应该编码所有特殊的字符，或者把所有内容嵌套在 CDATA 部分中。

为了确保在 XHTML 文档中脚本正确进行解析，请使用如下语法：

```
<script type="text/javascript"><![CDATA[
  document.write("Hello World!")
//]]></script>
```

必选的属性

属性	值	描述
type	*MIME-type*	指示脚本的 MIME 类型。

可选的属性

属性	值	描述
async	async	规定异步执行脚本（仅适用于外部脚本）。
charset	*charset*	规定在外部脚本文件中使用的字符编码。
defer	defer	规定是否对脚本执行进行延迟，直到页面加载为止。
language	*script*	不赞成使用。规定脚本语言。请使用 type 属性代替它。
src	*URL*	规定外部脚本文件的 URL。
xml:space	preserve	规定是否保留代码中的空白。

全局属性

<script> 标签支持 HTML 中的全局属性。

TIY 实例

插入一段脚本

本例演示如何将脚本插入 HTML 文档。

```
<html>

<body>

<script type="text/javascript">
document.write("<h1>Hello World!</h1>")
</script>

</body>

</html>
```

运行于不支持脚本的浏览器

本例演示如何对付不支持脚本的浏览器。

```
<!DOCTYPE html>
<html>
<body>

<script type="text/javascript">
document.write("Hello World!")
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscrip

<p>不支持 JavaScript 的浏览器将显示 noscript 元素中的文本。</p>

</body>
</html>
```

相关页面

教程：JavaScript 教程

教程：VBScript 教程

相关页面

HTML DOM 参考手册：Script 对象

HTML <section> 标签

实例

文档中的区段，解释了 PRC：

```
<section>
  <h1>PRC</h1>
  <p>The People's Republic of China was born in 1949...</p>
</section>
```

定义和用法

<section> 标签定义文档中的节（section、区段）。比如章节、页眉、页脚或文档中的其他部分。

HTML 4.01 与 HTML 5 之间的差异

<section> 标签是 HTML 5 中的新标签。

属性

属性	值	描述
cite	*URL*	section 的 URL，假如 section 摘自 web 的话。

全局属性

<section> 标签支持 HTML 中的全局属性。

事件属性

<section> 标签支持 HTML 中的事件属性。

HTML <select> 标签

实例

创建带有 4 个选项的选择列表：

```
<select>
  <option value ="volvo">Volvo</option>
  <option value ="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <select> 标签。

定义和用法

select 元素可创建单选或多选菜单。

<select&> 元素中的 <option> 标签用于定义列表中的可用选项。

HTML 4.01 与 HTML 5 之间的差异

HTML5 添加了一些新属性。

提示和注释：

提示：select 元素是一种表单控件，可用于在表单中接受用户输入。

属性

New: HTML5 中的新属性。

属性	值	描述
autofocus	autofocus	规定在页面加载后文本区域自动获得焦点。
disabled	disabled	规定禁用该下拉列表。
form	*form_id*	规定文本区域所属的一个或多个表单。
multiple	multiple	规定可选择多个选项。
name	*name*	规定下拉列表的名称。
required	required	规定文本区域是必填的。
size	*number*	规定下拉列表中可见选项的数目。

全局属性

<select> 标签支持 HTML 中的全局属性。

事件属性

<select> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Select 对象

HTML <small> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <small> 标签。

定义和用法

<small> 标签呈现小号字体效果。

<small> 标签和它所对应的 <big> 标签一样，但它是缩小字体而不是放大。如果被包围的字体已经是字体模型所支持的最小字号，那么 <small> 标签将不起任何作用。

与 <big> 标签类似，<small> 标签也可以嵌套，从而连续地把文字缩小。每个 <small> 标签都把文本的字体变小一号，直到达到下限的一号字。

参阅

请参阅 HTML 5 参考手册，了解有关 <small> 标签的最新信息：

HTML 5 <small> 标签

HTML <source> 标签

实例

拥有两份源文件的音频播放器。浏览器应该选择它所支持的文件（如果有的话）：

```
<audio controls>
   <source src="horse.ogg" type="audio/ogg">
   <source src="horse.mp3" type="audio/mpeg">
 Your browser does not support the audio element.
</audio>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <source> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <source> 标签。

定义和用法

<source> 标签为媒介元素（比如 <video> 和 <audio>）定义媒介资源。

<source> 标签允许您规定可替换的视频/音频文件供浏览器根据它对媒体类型或者编解码器的支持进行选择。

HTML 4.01 与 HTML 5 之间的差异

<source> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
media	*media query*	规定媒体资源的类型。
src	*url*	规定媒体文件的 URL。
type	*numeric value*	规定媒体资源的 MIME 类型。

全局属性

<source> 标签支持 HTML 中的全局属性。

事件属性

<source> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Source 对象

HTML <span> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <span> 标签。

定义和用法

<span> 标签被用来组合文档中的行内元素。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

提示：请使用 <span> 来组合行内元素，以便通过样式来格式化它们。

注释：span 没有固定的格式表现。当对它应用样式时，它才会产生视觉上的变化。

例子

```
<p><span>some text.</span>some other text.</p>
```

例子解释

如果不对 span 应用样式，那么 span 元素中的文本与其他文本不会任何视觉上的差异。尽管如此，上例中的 span 元素仍然为 p 元素增加了额外的结构。

可以为 span 应用 id 或 class 属性，这样既可以增加适当的语义，又便于对 span 应用样式。

可以对同一个 <span> 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种。这两者的主要差异是，class 用于元素组（类似的元素，或者可以理解为某一类元素），而 id 用于标识单独的唯一的元素。

提示：事实上，您也许已经注意到了，W3School 站点上有一些文本的样式与其他文本是不同的。比如"提示"使用了粗体的橘红色。尽管实现这种效果的方法非常多，但是我们的做法是：使用"提示"使用 span 元素，然后对这个 span 元素的父元素，即 p 元素应用 class，这样就可以对这个类的子元素 span 应用相应的样式了。

HTML:

```
<p class="tip"><span>提示：</span>... ... ...</p>
```

CSS:

```
p.tip span {
    font-weight:bold;
    color:#ff9955;
    }
```

全局属性

<span> 标签支持 HTML 中的全局属性。

事件属性

<span> 标签支持 HTML 中的事件属性。

HTML <strike> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <strike> 标签。

定义和用法

<strike> 标签可定义加删除线文本定义。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，<strike> 标签不被赞成使用。

在 XHTML 1.0 Strict DTD 中，<strike> 标签不被支持。

提示和注释：

注释：请使用 <del> 替代它！

标准属性

```
id, class, title, style, dir, lang, xml:lang
```

如需完整的描述，请访问标准属性。

事件属性

```
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
```

如需完整的描述，请访问事件属性。

TIY 实例

删除线文本

此例演示如何为文本标记添加删除线。

```
<html>

<body>

<p>在 HTML 5 中，<strike>仍然支持</strike>已经不支持这个标签了。</p>

</body>
</html>
```

参阅

请参阅 HTML 5 参考手册，了解有关 <strike> 标签的最新信息：

HTML 5 <strike> 标签

HTML <style> 标签

实例

```
<html>
<head>
<style type="text/css">
h1 {color:red}
p {color:blue}
</style>
</head>

<body>
<h1>Header 1</h1>
<p>A paragraph.</p>
</body>
</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <style> 标签。

定义和用法

<style> 标签用于为 HTML 文档定义样式信息。

在 style 中，您可以规定在浏览器中如何呈现 HTML 文档。

type 属性是必需的，定义 style 元素的内容。唯一可能的值是 "text/css"。

style 元素位于 head 部分中。

提示和注释

提示：如需链接外部样式表，请使用 <link> 标签。

提示：如需学习更多有关样式表的知识，请阅读我们的 CSS 教程。

HTML 与 XHTML 之间的差异

NONE

必需的属性

属性	值	描述
type	text/css	规定样式表的 MIME 类型。

可选的属性

属性	值	描述
media	`screen` `tty` `tv` `projection` `handheld` `print` `braille` `aural` `all`	为样式表规定不同的媒介类型。

全局属性

<style> 标签支持 HTML 中的全局属性。

事件属性

<style> 标签支持 HTML 中的事件属性。

TIY 实例

HTML中的样式

本例演示如何使用添加到 <head> 部分的样式信息对 HTML 进行格式化。

```
<html>

<head>
<style type="text/css">
h1 {color: red}
p {color: blue}
</style>
</head>

<body>
<h1>header 1</h1>
<p>A paragraph.</p>
</body>

</html>
```

没有下划线的链接

本例演示如何使用样式属性做一个没有下划线的链接。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312'
<meta http-equiv="Content-Language" content="zh-cn" />
</head>

<body>

<a href="/example/html/lastpage.html" style="text-decoration:none">
这是一个链接！
</a>

</body>
</html>
```

链接到一个外部样式表

本例演示如何<link>标签链接到一个外部样式表。

```
<html>

<head>
<link rel="stylesheet" type="text/css" href="/html/csstest1.css" >
</head>

<body>
<h1>我通过外部样式表进行格式化。</h1>
<p>我也一样！</p>
</body>

</html>
```

相关页面

HTML DOM 参考手册：Style 对象

HTML <sub> 标签

例子

```
这段文本包含 <sub>下标</sub>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <sub> 标签。

定义和用法

<sub> 标签可定义下标文本。

包含在 _{标签和其结束标签} 中的内容将会以当前文本流中字符高度的一半来显示，但是与当前文本流中文字的字体和字号都是一样的。

提示：无论是 <sub> 标签还是和它对应的 <sup> 标签，在数学等式、科学符号和化学公式中都非常有用。

HTML 与 XHTML 之间的差异

NONE

全局属性

<sub> 标签支持 HTML 中的全局属性。

事件属性

<sub> 标签支持 HTML 中的事件属性。

TIY 实例

上标和下标

本例演示如何使用 sup 和 sub 元素。

```
<html>
<body>

<p>
This text contains <sub>subscript</sub>
</p>

<p>
This text contains <sup>superscript</sup>
</p>

</body>
</html>
```

HTML <summary> 标签

实例

有关文档的详细信息：

```
<details>
<summary>HTML 5</summary>
This document teaches you everything you have to learn about HTML 5
</details>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

只有 Chrome 以及 Safari 6 支持 <summary> 标签。

定义和用法

<summary> 标签包含 details 元素的标题，"details" 元素用于描述有关文档或文档片段的详细信息。

HTML 4.01 与 HTML 5 之间的差异

<summary> 标签是 HTML 5 中的新标签。

提示和注释

提示：请与 <details> 标签一起使用。标题是可见的，当用户点击标题时会显示出详细信息。

注释："summary" 元素应该是 "details" 元素的第一个子元素。

全局属性

<summary> 标签支持 HTML 中的全局属性。

事件属性

<summary> 标签支持 HTML 中的事件属性。

HTML <sup> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <sup> 标签。

定义和用法

<sup> 标签可定义上标文本。

包含在 ^{标签和其结束标签} 中的内容将会以当前文本流中字符高度的一半来显示，但是与当前文本流中文字的字体和字号都是一样的。

提示：这个标签在向文档添加脚注以及表示方程式中的指数值时非常有用。如果和 <a> 标签结合起来使用，就可以创建出很好的超链接脚注。

HTML 与 XHTML 之间的差异

NONE

例子

```
这段文本包含 <sup>上标</sup>
```

全局属性

<sup> 标签支持 HTML 中的全局属性。

事件属性

<sup> 标签支持 HTML 中的事件属性。

HTML <table> 标签

实例

一个简单的 HTML 表格，包含两行两列：

```
<table border="1">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <table> 标签。

定义和用法

<table> 标签定义 HTML 表格。

简单的 HTML 表格由 table 元素以及一个或多个 tr、th 或 td 元素组成。

tr 元素定义表格行，th 元素定义表头，td 元素定义表格单元。

更复杂的 HTML 表格也可能包括 caption、col、colgroup、thead、tfoot 以及 tbody 元素。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，table 元素的 "align" 和 "bgcolor" 属性是不被赞成使用的。

在 XHTML 1.0 Strict DTD，table 元素的 "align" 和 "bgcolor" 属性是不被支持的。

可选的属性

属性	值	描述
align	`left` `center` `right`	不赞成使用。请使用样式代替。规定表格相对周围元素的对齐方式。
bgcolor	*rgb(x,x,x) #xxxxxx colorname*	不赞成使用。请使用样式代替。规定表格的背景颜色。
border	*pixels*	规定表格边框的宽度。
cellpadding	*pixels %*	规定单元边沿与其内容之间的空白。
cellspacing	*pixels %*	规定单元格之间的空白。
frame	`void` `above` `below` `hsides` `lhs` `rhs` `vsides` `box` `border`	规定外侧边框的哪个部分是可见的。
rules	`none` `groups` `rows` `cols` `all`	规定内侧边框的哪个部分是可见的。
summary	*text*	规定表格的摘要。
width	*% pixels*	规定表格的宽度。

全局属性

<table> 标签支持 HTML 中的全局属性。

事件属性

<table> 标签支持 HTML 中的事件属性。

TIY 实例

表格

这个例子演示如何在 HTML 文档中创建表格。

```
<html>

<body>

<p>每个表格由 table 标签开始。</p>
<p>每个表格行由 tr 标签开始。</p>
<p>每个表格数据由 td 标签开始。</p>

<h4>一列：</h4>
<table border="1">
<tr>
  <td>100</td>
</tr>
</table>

<h4>一行三列：</h4>
<table border="1">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
</table>

<h4>两行三列：</h4>
<table border="1">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

</body>
</html>
```

表格边框

本例演示各种类型的表格边框。

```
<html>

<body>

<h4>带有普通的边框：</h4>
<table border="1">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>带有粗的边框：</h4>
<table border="8">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>带有很粗的边框：</h4>
<table border="15">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

</body>
</html>
```

没有边框的表格

本例演示一个没有边框的表格。

```
<html>

<body>

<h4>这个表格没有边框：</h4>
<table>
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

<h4>这个表格也没有边框：</h4>
<table border="0">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

</body>
</html>
```

表格中的标题(Headings)

本例演示如何显示表格标题。

```
<html>

<body>

<h4>表头：</h4>
<table border="1">
<tr>
  <th>姓名</th>
  <th>电话</th>
  <th>电话</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
</table>

<h4>垂直的表头：</h4>
<table border="1">
<tr>
  <th>姓名</th>
  <td>Bill Gates</td>
</tr>
<tr>
  <th>电话</th>
  <td>555 77 854</td>
</tr>
<tr>
  <th>电话</th>
  <td>555 77 855</td>
</tr>
</table>

</body>
</html>
```

空单元格

本例展示如何使用 " " 处理没有内容的单元格。

```
<html>

<body>

<table border="1">
<tr>
  <td>Some text</td>
  <td>Some text</td>
</tr>
<tr>
  <td></td>
  <td>Some text</td>
</tr>
</table>

<p>正如您看到的，其中一个单元没有边框。这是因为它是空的。在该单元中插入一个空格

<p>我们的技巧是在单元中插入一个 no-breaking 空格。</p>

<p>no-breaking 空格是一个字符实体。如果您不清楚什么是字符实体，请阅读关于字符

<p>no-breaking 空格由和号开始 ("&")，然后是字符"nbsp"，并以分号结尾(";")。

</body>
</html>
```

带有标题的表格

本例演示一个带标题 (caption) 的表格

```
<html>

<body>

<h4>这个表格有一个标题，以及粗边框：</h4>

<table border="6">
<caption>我的标题</caption>
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

</body>
```

跨行或跨列的表格单元格

本例演示如何定义跨行或跨列的表格单元格。

```
<html>

<body>

<h4>横跨两列的单元格：</h4>
<table border="1">
<tr>
  <th>姓名</th>
  <th colspan="2">电话</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
</table>

<h4>横跨两行的单元格：</h4>
<table border="1">
<tr>
  <th>姓名</th>
  <td>Bill Gates</td>
</tr>
<tr>
  <th rowspan="2">电话</th>
  <td>555 77 854</td>
</tr>
<tr>
  <td>555 77 855</td>
</tr>
</table>

</body>
</html>
```

表格内的标签

本例演示如何在不同的元素内显示元素。

```
<html>

<body>

<table border="1">
<tr>
  <td>
   <p>这是一个段落。</p>
   <p>这是另一个段落。</p>
  </td>
  <td>这个单元包含一个表格：
   <table border="1">
   <tr>
     <td>A</td>
     <td>B</td>
   </tr>
   <tr>
     <td>C</td>
     <td>D</td>
   </tr>
   </table>
  </td>
</tr>
<tr>
  <td>这个单元包含一个列表：
   <ul>
    <li>苹果</li>
    <li>香蕉</li>
    <li>菠萝</li>
   </ul>
  </td>
  <td>HELLO</td>
</tr>
</table>

</body>
</html>
```

单元格填充(Cell padding)

本例演示如何使用单元格填充来创建单元格内容与其边框之间的空白。

```
<html>

<body>

<h4>没有 cellpadding：</h4>
<table border="1">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>带有 cellpadding：</h4>
<table border="1"
cellpadding="10">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

</body>
</html>
```

单元格间距(Cell spacing)

本例演示如何使用单元格间距增加单元格之间的距离。

```
<html>

<body>

<h4>没有 cellspacing：</h4>
<table border="1">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>带有 cellspacing：</h4>
<table border="1"
cellspacing="10">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

</body>
</html>
```

向表格添加背景颜色或背景图像

本例演示如何向表格添加背景

```
<html>

<body>

<h4>背景颜色：</h4>
<table border="1"
bgcolor="red">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

<h4>背景图像：</h4>
<table border="1"
background="/i/eg_bg_07.gif">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>

</body>
</html>
```

向表格单元添加背景颜色或者背景图像

本例演示如何向一个或者更多表格单元添加背景

```
#### 单元背景：

   First |
   Row |

  Second |
   Row |
```

在表格单元中排列内容

本例演示如何使用 "align" 属性排列单元格内容,以便创建一个美观的表格。

```
<html>

<body>

<table width="400" border="1">
 <tr>
  <th align="left">消费项目....</th>
  <th align="right">一月</th>
  <th align="right">二月</th>
 </tr>
 <tr>
  <td align="left">衣服</td>
  <td align="right">$241.10</td>
  <td align="right">$50.20</td>
 </tr>
 <tr>
  <td align="left">化妆品</td>
  <td align="right">$30.00</td>
  <td align="right">$44.45</td>
 </tr>
 <tr>
  <td align="left">食物</td>
  <td align="right">$730.40</td>
  <td align="right">$650.00</td>
 </tr>
 <tr>
  <th align="left">总计</th>
  <th align="right">$1001.50</th>
  <th align="right">$744.65</th>
 </tr>
</table>

</body>
</html>
```

框架(frame)属性

本例演示如何使用 "frame" 属性来控制围绕表格的边框。

```
<html>
<body>

<p><b>注释：</b>frame 属性无法在 Internet Explorer 中正确地显示。</p>

<p>Table with frame="box":</p>
<table frame="box">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
```

```
    <td>January</td>
    <td>$100</td>
  </tr>
</table>

<p>Table with frame="above":</p>
<table frame="above">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>

<p>Table with frame="below":</p>
<table frame="below">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>

<p>Table with frame="hsides":</p>
<table frame="hsides">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>

<p>Table with frame="vsides":</p>
<table frame="vsides">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```

```
</body>
</html>
```

相关页面

HTML DOM 参考手册：Table 对象

HTML <tbody> 标签

实例

带有 thead、tbody 以及 tfoot 元素的 HTML 表格：

```
<table border="1">
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>January</td>
      <td>$100</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$80</td>
    </tr>
  </tbody>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

<tbody> 标签仅得到所有主流浏览器的部分支持。

定义和用法

<tbody> 标签表格主体（正文）。该标签用于组合 HTML 表格的主体内容。

tbody 元素应该与 thead 和 tfoot 元素结合起来使用。

thead 元素用于对 HTML 表格中的表头内容进行分组，而 tfoot 元素用于对 HTML 表格中的表注（页脚）内容进行分组。

注释：如果您使用 thead、tfoot 以及 tbody 元素，您就必须使用全部的元素。它们的出现次序是：thead、tfoot、tbody，这样浏览器就可以在收到所有数据前呈现页脚了。您必须在 table 元素内部使用这些标签。

提示：在默认情况下这些元素不会影响到表格的布局。不过，您可以使用 CSS 使这些元素改变表格的外观。

详细描述

thead、tfoot 以及 tbody 元素使您有能力对表格中的行进行分组。当您创建某个表格时，您也许希望拥有一个标题行，一些带有数据的行，以及位于底部的一个总计行。这种划分使浏览器有能力支持独立于表格标题和页脚的表格正文滚动。当长的表格被打印时，表格的表头和页脚可被打印在包含表格数据的每张页面上。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：<thead> 内部必须拥有 <tr> 标签！

注释：<thead>、<tbody> 以及 <tfoot> 很少被使用，这是因为糟糕的浏览器支持。我们期望在 XHTML 的未来版本中这种情况会发生变化。假如您使用 Internet Explorer 5.0 或更新的浏览器，可以在我们的 XML 教程中查看一个例子。

可选的属性

属性	值	描述
align	`right` `left` `center` `justify` `char`	定义 tbody 元素中内容的对齐方式。
char	*character*	规定根据哪个字符来进行文本对齐。
charoff	*number*	规定第一个对齐字符的偏移量。
valign	`top` `middle` `bottom` `baseline`	规定 tbody 元素中内容的垂直对齐方式。

全局属性

<tbody> 标签支持 HTML 中的全局属性。

事件属性

<tbody> 标签支持 HTML 中的事件属性。

HTML <td> 标签

实例

一个简单的 HTML 表格，包含两行两列：

```
<table border="1">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <td> 标签。

定义和用法

<td> 标签定义 HTML 表格中的标准单元格。

HTML 表格有两类单元格：

- 表头单元 - 包含头部信息（由 th 元素创建）
- 标准单元 - 包含数据（由 td 元素创建）

td 元素中的文本一般显示为正常字体且左对齐。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，td 元素的 "bgcolor"、"height"、"width" 以及 "nowrap" 是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，td 元素的 "bgcolor"、"height"、"width" 以及 "nowrap" 是不被支持的。

提示和注释

提示：请使用 colspan 和 rowspan 属性来实现内容横跨多个行或列。

可选的属性

属性	值	描述
abbr	*text*	规定单元格中内容的缩写版本。
align	`left` `right` `center` `justify` `char`	规定单元格内容的水平对齐方式。
axis	*category_name*	对单元进行分类。
bgcolor	*rgb(x,x,x) #xxxxxx colorname*	不赞成使用。请使用样式取而代之。规定单元格的背景颜色。
char	*character*	规定根据哪个字符来进行内容的对齐。
charoff	*number*	规定对齐字符的偏移量。
colspan	*number*	规定单元格可横跨的列数。
headers	*headercells'id*	规定与单元格相关的表头。
height	*pixels %*	不赞成使用。请使用样式取而代之。规定表格单元格的高度。
nowrap	nowrap	不赞成使用。请使用样式取而代之。规定单元格中的内容是否折行。
rowspan	*number*	规定单元格可横跨的行数。
scope	`col` `colgroup` `row` `rowgroup`	定义将表头数据与单元数据相关联的方法。
valign	`top` `middle` `bottom` `baseline`	规定单元格内容的垂直排列方式。
width	*pixels %*	不赞成使用。请使用样式取而代之。规定表格单元格的宽度。

全局属性

<td> 标签支持 HTML 中的全局属性。

事件属性

<td> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：TableData 对象

HTML <textarea> 标签

实例

```
<textarea rows="3" cols="20">
在w3school，你可以找到你所需要的所有的网站建设教程。
</textarea>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <textarea> 标签。

定义和用法

<textarea> 标签定义多行的文本输入控件。

文本区中可容纳无限数量的文本，其中的文本的默认字体是等宽字体（通常是 Courier）。

可以通过 cols 和 rows 属性来规定 textarea 的尺寸，不过更好的办法是使用 CSS 的 height 和 width 属性。

注释：在文本输入区内的文本行间，用 "%OD%OA" （回车/换行）进行分隔。

提示：可以通过 <textarea> 标签的 wrap 属性设置文本输入区内的换行模式。有关 wrap 属性的详细信息。

HTML 与 XHTML 之间的差异

NONE

属性

New: HTML5 中的新属性。

属性	值	描述
autofocus	autofocus	规定在页面加载后文本区域自动获得焦点。
cols	*number*	规定文本区内的可见宽度。
disabled	disabled	规定禁用该文本区。
form	*form_id*	规定文本区域所属的一个或多个表单。
maxlength	*number*	规定文本区域的最大字符数。
name	*name_of_textarea*	规定文本区的名称。
placeholder	*text*	规定描述文本区域预期值的简短提示。
readonly	readonly	规定文本区为只读。
required	required	规定文本区域是必填的。
rows	*number*	规定文本区内的可见行数。
wrap	`hard` `soft`	规定当在表单中提交时，文本区域中的文本如何换行。。

全局属性

<textarea> 标签支持 HTML 中的全局属性。

事件属性

<textarea> 标签支持 HTML 中的事件属性。

HTML <tfoot> 标签

实例

带有 thead、tbody 以及 tfoot 元素的 HTML 表格：

```
<table border="1">
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>January</td>
      <td>$100</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$80</td>
    </tr>
  </tbody>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

<tfoot> 标签仅得到所有主流浏览器的部分支持。

定义和用法

<tfoot> 标签定义表格的页脚（脚注或表注）。该标签用于组合 HTML 表格中的表注内容。

tfoot 元素应该与 thead 和 tbody 元素结合起来使用。

thead 元素用于对 HTML 表格中的表头内容进行分组，而 tbody 元素用于对 HTML 表格中的主体内容进行分组。

注释：如果您使用 thead、tfoot 以及 tbody 元素，您就必须使用全部的元素。它们的出现次序是：thead、tfoot、tbody，这样浏览器就可以在收到所有数据前呈现页脚了。您必须在 table 元素内部使用这些标签。

提示：在默认情况下这些元素不会影响到表格的布局。不过，您可以使用 CSS 使这些元素改变表格的外观。

详细描述

thead、tfoot 以及 tbody 元素使您有能力对表格中的行进行分组。当您创建某个表格时，您也许希望拥有一个标题行，一些带有数据的行，以及位于底部的一个总计行。这种划分使浏览器有能力支持独立于表格标题和页脚的表格正文滚动。当长的表格被打印时，表格的表头和页脚可被打印在包含表格数据的每张页面上。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：<thead> 内部必须拥有 <tr> 标签！

注释：<thead>、<tbody> 以及 <tfoot> 很少被使用，这是因为糟糕的浏览器支持。我们期望在 XHTML 的未来版本中这种情况会发生变化。假如您使用 Internet Explorer 5.0 或更新的浏览器，可以在我们的 XML 教程中查看一个例子。

可选的属性

属性	值	描述
align	`right` `left` `center` `justify` `char`	定义 tfoot 元素中内容的对齐方式。
char	*character*	规定根据哪个字符来进行文本对齐。
charoff	*number*	规定第一个对齐字符的偏移量。
valign	`top` `middle` `bottom` `baseline`	规定 tfoot 元素中内容的垂直对齐方式。

全局属性

<tfoot> 标签支持 HTML 中的全局属性。

事件属性

<tfoot> 标签支持 HTML 中的事件属性。

HTML <th> 标签

实例

普通的 HTML 表格，包含两行两列：

```
<table border="1">
  <tr>
    <th>Company</th>
    <th>Address</th>
  </tr>

  <tr>
    <td>Apple, Inc.</td>
    <td>1 Infinite Loop Cupertino, CA 95014</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <th> 标签。

定义和用法

定义表格内的表头单元格。

HTML 表单中有两种类型的单元格：

- 表头单元格 - 包含表头信息（由 th 元素创建）
- 标准单元格 - 包含数据（由 td 元素创建）

th 元素内部的文本通常会呈现为居中的粗体文本，而 td 元素内的文本通常是左对齐的普通文本。

提示和注释

提示：如果需要将内容横跨多个行或列，请使用 colspan 和 rowspan 属性。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，th 元素的 "bgcolor"、"height"、"width" 以及 "nowrap" 属性是不被推荐使用的。

在 XHTML 1.0 Strict DTD 中，th 元素的 "bgcolor"、"height"、"width" 以及 "nowrap" 属性是不被支持。

可选的属性

属性	值	描述
abbr	*text*	规定单元格中内容的缩写版本。
align	`left` `right` `center` `justify` `char`	规定单元格内容的水平对齐方式。
axis	*category_name*	对单元格进行分类。
bgcolor	*rgb(x,x,x) #xxxxxx colorname*	不推荐使用。请使用样式替代它。规定表格单元格的背景颜色。
char	*character*	规定根据哪个字符来进行内容的对齐。
charoff	*number*	规定对齐字符的偏移量。
colspan	*number*	设置单元格可横跨的列数。
headers	*idrefs*	由空格分隔的表头单元格 ID 列表，为数据单元格提供表头信息。
height	*pixels %*	不推荐使用。请使用样式替代它。规定表格单元格的高度。
nowrap	nowrap	不推荐使用。请使用样式取而代之。规定单元格中的内容是否折行。
rowspan	*number*	规定单元格可横跨的行数。
scope	`col` `colgroup` `row` `rowgroup`	定义将表头数据与单元数据相关联的方法。
valign	`top` `middle` `bottom` `baseline`	规定单元格内容的垂直排列方式。
width	*pixels %*	不推荐使用。请使用样式取而代之。规定表格单元格的宽度。

全局属性

<th> 标签支持 HTML 中的全局属性。

事件属性

<th> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：TableHeader 对象

HTML <thead> 标签

实例

带有 thead、tbody 以及 tfoot 元素的 HTML 表格：

```
<table border="1">
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>January</td>
      <td>$100</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$80</td>
    </tr>
  </tbody>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

<thead> 标签仅得到所有主流浏览器的部分支持。

定义和用法

<thead> 标签定义表格的表头。该标签用于组合 HTML 表格的表头内容。

thead 元素应该与 tbody 和 tfoot 元素结合起来使用。

tbody 元素用于对 HTML 表格中的主体内容进行分组，而 tfoot 元素用于对 HTML 表格中的表注（页脚）内容进行分组。

注释：如果您使用 thead、tfoot 以及 tbody 元素，您就必须使用全部的元素。它们的出现次序是：thead、tfoot、tbody，这样浏览器就可以在收到所有数据前呈现页脚了。您必须在 table 元素内部使用这些标签。

提示：在默认情况下这些元素不会影响到表格的布局。不过，您可以使用 CSS 使这些元素改变表格的外观。

详细描述

thead、tfoot 以及 tbody 元素使您有能力对表格中的行进行分组。当您创建某个表格时，您也许希望拥有一个标题行，一些带有数据的行，以及位于底部的一个总计行。这种划分使浏览器有能力支持独立于表格标题和页脚的表格正文滚动。当长的表格被打印时，表格的表头和页脚可被打印在包含表格数据的每张页面上。

HTML 与 XHTML 之间的差异

NONE

提示和注释：

注释：<thead> 内部必须拥有 <tr> 标签！

注释：<thead>、<tbody> 以及 <tfoot> 很少被使用，这是因为糟糕的浏览器支持。我们期望在 XHTML 的未来版本中这种情况会发生变化。假如您使用 Internet Explorer 5.0 或更新的浏览器，可以在我们的 XML 教程中查看一个例子。

可选的属性

属性	值	描述
align	`right` `left` `center` `justify` `char`	定义 thead 元素中内容的对齐方式。
char	*character*	规定根据哪个字符来进行文本对齐。
charoff	*number*	规定第一个对齐字符的偏移量。
valign	`top` `middle` `bottom` `baseline`	规定 thead 元素中内容的垂直对齐方式。

全局属性

<thead> 标签支持 HTML 中的全局属性。

事件属性

<thead> 标签支持 HTML 中的事件属性。

HTML <time> 标签

实例

如何定义时间和日期：

```
<p>我们在每天早上 <time>9:00</time> 开始营业。</p>

<p>我在 <time datetime="2008-02-14">情人节</time> 有个约会。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

<time> 标签不会在任何浏览器中呈现任何特殊效果。

定义和用法

<time> 标签定义公历的时间（24 小时制）或日期，时间和时区偏移是可选的。

该元素能够以机器可读的方式对日期和时间进行编码，这样，举例说，用户代理能够把生日提醒或排定的事件添加到用户日程表中，搜索引擎也能够生成更智能的搜索结果。

浏览器支持

目前所有主流浏览器都不支持 <time> 标签。

HTML 4.01 与 HTML 5 之间的差异

<time> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
datetime	*datetime*	规定日期 / 时间。否则，由元素的内容给定日期 / 时间。
pubdate	*pubdate*	指示 <time> 元素中的日期 / 时间是文档（或 <article> 元素）的发布日期。

全局属性

<time> 标签支持 HTML 中的全局属性。

事件属性

<time> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Time 对象

HTML <title> 标签

实例

一个简单的 HTML 文档，带有尽可能少的必需的标签：

```
<html>

  <head>
    <title>XHTML Tag Reference</title>
  </head>

  <body>
    The content of the document......
  </body>

</html>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <title> 标签。

定义和用法

<title> 元素可定义文档的标题。

浏览器会以特殊的方式来使用标题，并且通常把它放置在浏览器窗口的标题栏或状态栏上。同样，当把文档加入用户的链接列表或者收藏夹或书签列表时，标题将成为该文档链接的默认名称。

HTML 与 XHTML 之前的差异

NONE

提示和注释

提示：<title> 标签是 <head> 标签中唯一要求包含的东西。

标准属性

<title> 标签支持以下标准属性：

属性	值	描述
dir	`rtl` `ltr`	规定元素中内容的文本方向。
lang	*language_code*	规定元素中内容的语言代码。
xml:lang	*language_code*	规定 XHTML 文档中元素内容的语言代码。

DTD：指示在哪种 XHTML 1.0 DTD 中允许该标签。S=Strict, T=Transitional, F=Frameset.

如需更多有关 HTML 标准属性的信息，请访问标准属性。

全局属性

<title> 标签支持 HTML 中的全局属性。

延伸阅读：标题里是什么？

一定要选择一个正确的标题，这对于定义文档并确保它能够在 Web 上有效利用来说是十分重要的。

请记住，用户可以用任何顺序、独立地访问文档集中的每一个文档。所以，文档的标题不仅应当根据其他文档的上下文定义，而且还要显示其自身的特点。

含有文档引用排序的标题通常不是什么好标题。举个例子，像“第十六章”或“第五部分”这样的标题，对读者理解其内容方面毫无用处。描述性更强的标题，像“第十六章：HTML 标题”，或者“第五部分：如何使用标题”，这样的标题不仅表达了它在一个大型文档集中的位置，还说明了文档的具体内容，吸引读者更有兴趣读下去。

自我引用的标题也没有什么用处。像“主页”这样的标题和内容毫无关系，类似的还有“反馈页”或“常用链接”等。你应该设计一个能够传达一定内容和目的的标题，令读者凭这个标题就可以判断是否由必要访问这个页面。“HTML <title> 标签的详细信息”，这就是一个描述性的标题，类似的还有“HTML <title> 标签的反馈页”等等。

人们常常会花费大量时间去创建 Web 文档，但却经常只是因为一个不吸引人或无意义的标题，而把这些努力全都浪费掉了。当自动为用户搜集链接的特殊软件在 Web 上越来越流行时，只有网页的标题才会作为与页面相关的描述性词语，被插入到庞大的链接数据库中。因此，我们怎么强调这一点都不过分：请为自己的每个文档都认真地选择一个描述性的、实用的并与上下文独立的标题。

相关页面

HTML DOM 参考手册：Title 对象

HTML <tr> 标签

实例

一个简单的 HTML 表格，包含两行两列：

```
<table border="1">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <tr> 标签。

定义和用法

<tr> 标签定义 HTML 表格中的行。

tr 元素包含一个或多个 th 或 td 元素。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，tr 元素的 "bgcolor" 是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，tr 元素的 "bgcolor" 是不被支持的。

可选的属性

属性	值	描述
align	`right` `left` `center` `justify` `char`	定义表格行的内容对齐方式。
bgcolor	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取而代之。规定表格行的背景颜色。
char	*character*	规定根据哪个字符来进行文本对齐。
charoff	*number*	规定第一个对齐字符的偏移量。
valign	`top` `middle` `bottom` `baseline`	规定表格行中内容的垂直对齐方式。

全局属性

<tr> 标签支持 HTML 中的全局属性。

事件属性

<tr> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：TableRow 对象

HTML <track> 标签

实例

播放带有字幕的视频：

```
<video width="320" height="240" controls="controls">
  <source src="forrest_gump.mp4" type="video/mp4" />
  <source src="forrest_gump.ogg" type="video/ogg" />
  <track kind="subtitles" src="subs_chi.srt" srclang="zh" label="Ch
  <track kind="subtitles" src="subs_eng.srt" srclang="en" label="En
</video>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10, Chrome 以及 Opera 支持 <track> 标签。

定义和用法

<track> 标签为诸如 video 元素之类的媒介规定外部文本轨道。

用于规定字幕文件或其他包含文本的文件，当媒介播放时，这些文件是可见的。

浏览器支持

目前所有主流浏览器都不支持 <track> 标签。

HTML 4.01 与 HTML 5 之间的差异

<track> 标签是 HTML 5 中的新标签。

属性

new : HTML5 中的新属性。

属性	值	描述
default	default	规定该轨道是默认的，假如没有选择任何轨道。
kind	`captions` `chapters` `descriptions` `metadata` `subtitles`	表示轨道属于什么文本类型。
label	*label*	轨道的标签或标题。
src	*url*	轨道的 URL。
srclang	*language_code*	轨道的语言，若 kind 属性值是 "subtitles"，则该属性必需的。

全局属性

<track> 标签支持 HTML 中的全局属性。

事件属性

<track> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Track 对象

HTML <tt> 标签

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <tt> 标签。

定义和用法

<tt> 标签呈现类似打字机或者等宽的文本效果。

<tt> 标签与 <code> 和 <kbd> 标签一样，<tt> 标签和必需的 </tt> 结束标签告诉浏览器，要把其中包含的文本显示为等宽字体。对于那些已经使用了等宽字体的浏览器来说，这个标签在文本的显示上就没有什么特殊效果了。

参阅

请参阅 HTML 5 参考手册，了解有关 <tt> 标签的最新信息：

HTML 5 <tt> 标签

HTML <u> 标签

实例

使用 <u> 标签为文本添加下划线：

```
<p>如果文本不是超链接，就不要<u>对其使用下划线</u>。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <u> 标签。

定义和用法

<u> 标签可定义下划线文本。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，u 元素是不被推荐使用的。

在 XHTML 1.0 Strict DTD 中，u 元素是不被支持的。

提示和注释：

注释：请尽量避免为文本加下划线 - 用户会把它混淆为一个超链接！

全局属性

<u> 标签支持 HTML 中的全局属性。

事件属性

<u> 标签支持 HTML 中的事件属性。

HTML <ul> 标签

实例

无序 HTML 列表：

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

（请在页面底部查看更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 <ul> 标签。

定义和用法

 标签定义无序列表。

HTML 与 XHTML 之间的差异

在 HTML 4.01 中，ul 元素的 "compact" 和 "type" 属性是不被赞成使用的。

在 XHTML 1.0 Strict DTD 中，ul 元素的 "compact" 和 "type" 属性是不被支持的。

提示和注释：

提示：请使用样式来定义列表的类型。

可选的属性

属性	值	描述
compact	compact	不赞成使用。请使用样式取代它。规定列表呈现的效果比正常情况更小巧。
type	disc square circle	不赞成使用。请使用样式取代它。规定列表的项目符号的类型。

全局属性

 标签支持 HTML 中的全局属性。

事件属性

 标签支持 HTML 中的事件属性。

HTML <video> 标签

实例

一段简单的 HTML5 视频：

```
<video src="movie.ogg" controls="controls">
您的浏览器不支持 video 标签。
</video>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome 以及 Safari 支持 <video> 标签。

注释：Internet Explorer 8 以及更早的版本不支持 <video> 标签。

定义和用法

<video> 标签定义视频，比如电影片段或其他视频流。

HTML 4.01 与 HTML 5 之间的差异

<video> 标签是 HTML 5 的新标签。

提示和注释

提示：可以在开始标签和结束标签之间放置文本内容，这样老的浏览器就可以显示出不支持该标签的信息。

属性

new : HTML5 中的新属性。

属性	值	描述
autoplay	autoplay	如果出现该属性，则视频在就绪后马上播放。
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮。
height	*pixels*	设置视频播放器的高度。
loop	loop	如果出现该属性，则当媒介文件完成播放后再次开始播放。
muted	muted	规定视频的音频输出应该被静音。
poster	*URL*	规定视频下载时显示的图像，或者在用户点击播放按钮前显示的图像。
preload	preload	如果出现该属性，则视频在页面加载时进行加载，并预备播放。如果使用 "autoplay"，则忽略该属性。
src	*url*	要播放的视频的 URL。
width	*pixels*	设置视频播放器的宽度。

全局属性

<video> 标签支持 HTML 中的全局属性。

事件属性

<video> 标签支持 HTML 中的事件属性。

相关页面

HTML DOM 参考手册：Video 对象

HTML <wbr> 标签

实例

一段带有 Word Break Opportunity 的文本：

```
<p>
如果想学习 AJAX，那么您必须熟悉 XML<wbr>Http<wbr>Request 对象。
</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 <wbr> 标签，除了 Internet Explorer。

定义和用法

Word Break Opportunity (<wbr>) 规定在文本中的何处适合添加换行符。

提示：如果单词太长，或者您担心浏览器会在错误的位置换行，那么您可以使用 <wbr> 元素来添加 Word Break Opportunity（单词换行时机）。

浏览器支持

所有主流浏览器均支持 <wbr> 标签，除了 Internet Explorer。

HTML 4.01 与 HTML 5 之间的差异

<wbr> 标签是 HTML 5 中的新标签。

全局属性

<wbr> 标签支持 HTML 中的全局属性。

事件属性

<wbr> 标签支持 HTML 中的事件属性。

HTML 全局属性

HTML 属性赋予元素意义和语境。

下面的全局属性可用于任何 HTML 元素。

HTML 全局属性

= HTML5 中添加的属性。

属性	描述
accesskey	规定激活元素的快捷键。
class	规定元素的一个或多个类名（引用样式表中的类）。
contenteditable	规定元素内容是否可编辑。
contextmenu	规定元素的上下文菜单。上下文菜单在用户点击元素时显示。
data-*	用于存储页面或应用程序的私有定制数据。
dir	规定元素中内容的文本方向。
draggable	规定元素是否可拖动。
dropzone	规定在拖动被拖动数据时是否进行复制、移动或链接。
hidden	规定元素仍未或不再相关。
id	规定元素的唯一 id。
lang	规定元素内容的语言。
spellcheck	规定是否对元素进行拼写和语法检查。
style	规定元素的行内 CSS 样式。
tabindex	规定元素的 tab 键次序。
title	规定有关元素的额外信息。
translate	规定是否应该翻译元素内容。

HTML accesskey 属性

HTML 全局属性

实例

带有指定快捷键的超链接：

```
<a href="http://www.w3school.com.cn/html/" accesskey="h">HTML</a><b
<a href="http://www.w3school.com.cn/css/" accesskey="c">CSS</a>
```

浏览器支持

几乎所有浏览器均 accesskey 属性，除了 Opera。

定义和用法

accesskey 属性规定激活（使元素获得焦点）元素的快捷键。

提示和注释

注释：以下元素支持 accesskey 属性：<a>, <area>, <button>, <input>, <label>, <legend> 以及 <textarea>。

语法

```
<element accesskey="_character_">
```

属性值

值	描述
character	规定激活（使元素获得焦点）元素的便捷按键。

HTML 全局属性

HTML class 属性

HTML 全局属性

实例

在 HTML 文档中使用 class 属性：

```
<html>
<head>
<style type="text/css">
h1.intro {color:blue;}
p.important {color:green;}
</style>
</head>

<body>
<h1 class="intro">Header 1</h1>
<p>A paragraph.</p>
<p class="important">Note that this is an important paragraph.</p>
</body>
</html>
```

（可以在本页底部找到更多实例）

定义和用法

class 属性规定元素的类名（classname）。

class 属性大多数时候用于指向样式表中的类（class）。不过，也可以利用它通过 JavaScript 来改变带有指定 class 的 HTML 元素。

提示和注释

注释：class 属性不能在以下 HTML 元素中使用：base, head, html, meta, param, script, style 以及 title。

提示：可以给 HTML 元素赋予多个 class，例如：<span class="left_menu important">。这么做可以把若干个 CSS 类合并到一个 HTML 元素。

提示：类名不能以数字开头！只有 Internet Explorer 支持这种做法。

浏览器支持

W3C: "W3C" 列指示 W3C 的 HTML/XHTML 推荐标准中是否定义了该属性。

IE	Firefox	Opera	Safari	W3C
YES	YES	YES	YES	YES

语法

```
<element class="_value_">
```

属性值

值	描述
classname	规定元素的类的名称。如需为一个元素规定多个类，用空格分隔类名。

亲自试一试 - 实例

向元素添加多个类

如何向一个元素添加多个类。

HTML 全局属性

HTML contenteditable 属性

HTML 全局属性

实例

一段可编辑的段落：

```
<p contenteditable="true">这是一个可编辑的段落。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 contenteditable 属性。

定义和用法

contenteditable 属性规定元素内容是否可编辑。

注释：如果元素未设置 contenteditable 属性，那么元素会从其父元素继承该属性。

HTML 4.01 与 HTML5 之间的差异

contenteditable 属性是 HTML5 中的新属性。

语法

```
<_element_ contenteditable="true|false">
```

属性值

值	描述
true	规定元素可编辑。
false	规定元素不可编辑。

HTML 全局属性

HTML contextmenu 属性

HTML 全局属性

实例

规定 <div> 元素的上下文菜单。上下文菜单会在用户右键点击元素时出现：

```
<div contextmenu="mymenu">

<menu type="context" id="mymenu">
  <menuitem label="Refresh"></menuitem>
  <menuitem label="Twitter"></menuitem>
</menu>

</div>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前只有 Firefox 支持 contextmenu 属性。

定义和用法

contextmenu 属性规定元素的上下文菜单。当用户右键点击元素时，会出现上下文菜单。

contextmenu 属性的值是要打开的 <menu> 元素的 id。

HTML 4.01 与 HTML5 之间的差异

contextmenu 属性是 HTML5 中的新属性。

语法

```
<_element_ contextmenu="*menu_id*">
```

属性值

值	描述
menu_id	要打开的 <menu> 元素的 id。

HTML 全局属性

HTML data-* 属性

HTML 全局属性

实例

使用 data-* 属性来嵌入自定义数据：

```
<ul>
<li data-animal-type="bird">Owl</li>
<li data-animal-type="fish">Salmon</li>
<li data-animal-type="spider">Tarantula</li>
</ul>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 data-* 属性。

定义和用法

data-* 属性用于存储页面或应用程序的私有自定义数据。

data-* 属性赋予我们在所有 HTML 元素上嵌入自定义 data 属性的能力。

存储的（自定义）数据能够被页面的 JavaScript 中利用，以创建更好的用户体验（不进行 Ajax 调用或服务器端数据库查询）。

data-* 属性包括两部分：

- 属性名不应该包含任何大写字母，并且在前缀 "data-" 之后必须有至少一个字符
- 属性值可以是任意字符串

注释：用户代理会完全忽略前缀为 "data-" 的自定义属性。

HTML 4.01 与 HTML5 之间的差异

data-* 属性是 HTML5 中的新属性。

语法

```
<_element_ data-*="_somevalue_">
```

属性值

值	描述
somevalue	规定属性的值（以字符串）。

HTML 全局属性

HTML dir 属性

HTML 全局属性

实例

一段方向从右向左的段落：

```
<p dir="rtl">Write this text right-to-left!</p>
```

浏览器支持

所有浏览器均支持 dir 属性。

定义和用法

dir 属性规定元素内容的文本方向。

提示和注释

注释：dir 属性在以下标签中无效：<base>,
, <frame>, <frameset>, <hr>, <iframe>, <param> 以及 <script>。

语法

```
<element dir="ltr|rtl">
```

属性值

值	描述
ltr	默认。从左向右的文本方向。
rtl	从右向左的文本方向。

HTML 全局属性

HTML draggable 属性

HTML 全局属性

实例

一个可拖动的段落：

```
<p draggable="true">这是一个可拖动的段落。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari 支持 draggable 属性。

注释：Internet Explorer 8 以及更早的版本，不支持 draggable 属性。

定义和用法

draggable 属性规定元素是否可拖动。

提示：链接和图像默认是可拖动的。

提示：draggable 属性常用在拖放操作中。请在我们的拖放教程中学习更多内容。

HTML 4.01 与 HTML5 之间的差异

draggable 属性是 HTML5 中的新属性。

语法

```
<_element_ draggable="true|false|auto">
```

属性值

值	描述
true	规定元素的可拖动的。
false	规定元素不可拖动。
auto	使用浏览器的默认行为。

HTML 全局属性

HTML dropzone 属性

HTML 全局属性

实例

拖动数据会产生被拖动数据的副本：

```
<div dropzone="copy"></div>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前所有主流浏览器都不支持 contenteditable 属性。

定义和用法

dropzone 属性规定在元素上拖动数据时，是否拷贝、移动或链接被拖动数据。

HTML 4.01 与 HTML5 之间的差异

dropzone 属性是 HTML5 中的新属性。

语法

```
<_element_ dropzone="copy|move|link">
```

属性值

值	描述
copy	拖动数据会产生被拖动数据的副本。
move	拖动数据会导致拖动数据被移动到新位置。
link	拖动数据会产生指向原始数据的链接。

HTML 全局属性

HTML hidden 属性

HTML 全局属性

实例

被隐藏的段落：

```
<p hidden>这个段落应该被隐藏。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 hidden 属性，除了 Internet Explorer。

定义和用法

hidden 属性是布尔属性。

如果设置该属性，它规定元素仍未或不再相关。

浏览器不应显示已规定 hidden 属性的元素。

hidden 属性也可用于防止用户查看元素，直到匹配某些条件（比如选择了某个复选框）。然后，JavaScript 可以删除 hidden 属性，以使此元素可见。

HTML 4.01 与 HTML5 之间的差异

hidden 属性是 HTML5 中的新属性。

HTML 与 XHTML 之间的差异

在 XHTML 中，属性简写是禁止的，hidden 属性必须定义为 <element hidden="hidden">。

语法

```
<_element_ hidden>
```

HTML 全局属性

HTML id 属性

HTML 全局属性

实例

通过 JavaScript 利用 id 属性来改变一段文本：

```
<html>
<head>
<script type="text/javascript">
function change_header()
{
document.getElementById("myHeader").innerHTML="Nice day!";
}
</script>
</head>

<body>
<h1 id="myHeader">Hello World!</h1>
<button onclick="change_header()">Change text</button>
</body>

</html>
```

（可以在本页底部找到更多实例）

定义和用法

id 属性规定 HTML 元素的唯一的 id。

id 在 HTML 文档中必须是唯一的。

id 属性可用作链接锚（link anchor），通过 JavaScript（HTML DOM）或通过 CSS 为带有指定 id 的元素改变或添加样式。

支持

W3C: "W3C" 列指示 W3C 的 HTML/XHTML 推荐标准中是否定义了该属性。

IE	Firefox	Opera	Safari	W3C
YES	YES	YES	YES	YES

语法

```
<element id="_value_">
```

属性值

值	描述
id	规定元素的唯一 id。

亲自试一试 - 实例

向 HTML 元素添加唯一的 id

如何向 HTML 元素添加 id 属性。

在 CSS 中使用 id 属性

如何在 CSS 中使用 id 属性

HTML 全局属性

HTML lang 属性

HTML 全局属性

实例

段落中的一些法文：

```
<p lang="fr">Ceci est un paragraphe.</p>
```

浏览器支持

所有浏览器均支持 lang 属性。

定义和用法

lang 属性规定元素内容的语言。

提示和注释

注释：lang 属性在以下标签中无效：<base>,
, <frame>, <frameset>, <hr>, <iframe>, <param> 以及 <script>。

语法

```
<element lang="*language_code*">
```

属性值

值	描述
language_code	规定元素内容的语言代码。语言代码参考

HTML 全局属性

HTML spellcheck 属性

HTML 全局属性

实例

进行拼写检查的可编辑段落：

```
<p contenteditable="true" spellcheck="true">这是一个段落。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10, Firefox, Opera, Chrome 以及 Safari 支持 spellcheck 属性。

注释：Internet Explorer 9 以及更早的版本不支持 spellcheck 属性。

定义和用法

spellcheck 属性规定是否对元素进行拼写和语法检查。

可以对以下内容进行拼写检查：

- input 元素中的文本值（非密码）
- <textarea> 元素中的文本
- 可编辑元素中的文本

HTML 4.01 与 HTML5 之间的差异

spellcheck 属性是 HTML5 中的新属性。

语法

```
<_element_ spellcheck="true|false">
```

属性值

值	描述
true	对元素进行拼写和语法检查
false	不检查元素。

HTML 全局属性

HTML style 属性

HTML 全局属性

实例

在 HTML 文档中使用 style 属性：

```
<h1 style="color:blue; text-align:center">This is a header</h1>
<p style="color:red">This is a paragraph.</p>
```

（可以在本页底部找到更多实例）

定义和用法

style 属性规定元素的行内样式（inline style）

style 属性将覆盖任何全局的样式设定，例如在 <style> 标签或在外部样式表中规定的样式。

支持

W3C: "W3C" 列指示 W3C 的 HTML/XHTML 推荐标准中是否定义了该属性。

IE	Firefox	Opera	Safari	W3C
YES	YES	YES	YES	YES

语法

```
<element style="_value_">
```

属性值

值	描述
style_definition	一个或多个由分号分隔的 CSS 属性和值。

HTML 全局属性

HTML tabindex 属性

HTML 全局属性

实例

带有指定 tab 键顺序的链接：

```
<a href="http://www.w3school.com.cn/" tabindex="2">W3School</a>
<a href="http://www.google.com/" tabindex="1">Google</a>
<a href="http://www.microsoft.com/" tabindex="3">Microsoft</a>
```

浏览器支持

几乎所有浏览器均 tabindex 属性，除了 Safari。

定义和用法

tabindex 属性规定元素的 tab 键控制次序（当 tab 键用于导航时）。

提示和注释

注释：以下元素支持 tabindex 属性：<a>, <area>, <button>, <input>, <object>, <select> 以及 <textarea>。

语法

```
<element tabindex="_number_">
```

属性值

值	描述
number	规定元素的 tab 键控制次序（1 是第一个）。

HTML 全局属性

HTML title 属性

HTML 全局属性

实例

在 HTML 文档中使用 title 属性：

```
<abbr title="People's Republic of China">PRC</abbr> was founded in
<p title="Free Web tutorials">W3School.com.cn</p>
```

（可以在本页底部找到更多实例）

定义和用法

title 属性规定关于元素的额外信息。

这些信息通常会在鼠标移到元素上时显示一段工具提示文本（tooltip text）。

提示：title 属性常与 form 以及 a 元素一同使用，以提供关于输入格式和链接目标的信息。同时它也是 abbr 和 acronym 元素的必需属性。

支持

W3C: "W3C" 列指示 W3C 的 HTML/XHTML 推荐标准中是否定义了该属性。

IE	Firefox	Opera	Safari	W3C
YES	YES	YES	YES	YES

语法

```
<element title="_value_">
```

属性值

值	描述
text	规定元素的工具提示文本（tooltip text）。

HTML 全局属性

HTML translate 属性

HTML 全局属性

实例

规定不应翻译某些元素：

```
<p translate="no">请勿翻译本段。</p>
<p>本段可被译为任意语言。</p>
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都无法正确地支持 translate 属性。

定义和用法

translate 规定是否应该翻译元素内容。

提示：请使用 class="notranslate" 替代。

HTML 4.01 与 HTML5 之间的差异

translate 属性是 HTML5 中的新属性。

语法

```
<_element_ translate="yes|no">
```

属性值

值	描述
yes	规定应该翻译元素内容。
no	规定不应翻译元素内容。

HTML 全局属性

HTML 事件属性

全局事件属性

HTML 4 增加了使事件在浏览器中触发动作的能力，比如当用户点击元素时启动 JavaScript。

如需学习更多有关事件编程的知识，请访问我们的 JavaScript 教程。

下面列出了添加到 HTML 元素以定义事件动作的全局事件属性。

= HTML5 中新的事件属性。

Window 事件属性

针对 window 对象触发的事件（应用到 <body> 标签）：

属性	值	描述
onafterprint	script	文档打印之后运行的脚本。
onbeforeprint	script	文档打印之前运行的脚本。
onbeforeunload	script	文档卸载之前运行的脚本。
onerror	script	在错误发生时运行的脚本。
onhaschange	script	当文档已改变时运行的脚本。
onload	script	页面结束加载之后触发。
onmessage	script	在消息被触发时运行的脚本。
onoffline	script	当文档离线时运行的脚本。
ononline	script	当文档上线时运行的脚本。
onpagehide	script	当窗口隐藏时运行的脚本。
onpageshow	script	当窗口成为可见时运行的脚本。
onpopstate	script	当窗口历史记录改变时运行的脚本。
onredo	script	当文档执行撤销（redo）时运行的脚本。
onresize	script	当浏览器窗口被调整大小时触发。
onstorage	script	在 Web Storage 区域更新后运行的脚本。
onundo	script	在文档执行 undo 时运行的脚本。
onunload	script	一旦页面已下载时触发（或者浏览器窗口已被关闭）。

Form 事件

由 HTML 表单内的动作触发的事件（应用到几乎所有 HTML 元素，但最常用在 form 元素中）：

属性	值	描述
onblur	script	元素失去焦点时运行的脚本。
onchange	script	在元素值被改变时运行的脚本。
oncontextmenu	script	当上下文菜单被触发时运行的脚本。
onfocus	script	当元素失去焦点时运行的脚本。
onformchange	script	在表单改变时运行的脚本。
onforminput	script	当表单获得用户输入时运行的脚本。
oninput	script	当元素获得用户输入时运行的脚本。
oninvalid	script	当元素无效时运行的脚本。
onreset	script	当表单中的重置按钮被点击时触发。HTML5 中不支持。
onselect	script	在元素中文本被选中后触发。
onsubmit	script	在提交表单时触发。

Keyboard 事件

属性	值	描述
onkeydown	script	在用户按下按键时触发。
onkeypress	script	在用户敲击按钮时触发。
onkeyup	script	当用户释放按键时触发。

Mouse 事件

由鼠标或类似用户动作触发的事件：

属性	值	描述
onclick	script	元素上发生鼠标点击时触发。
ondblclick	script	元素上发生鼠标双击时触发。
ondrag	script	元素被拖动时运行的脚本。
ondragend	script	在拖动操作末端运行的脚本。
ondragenter	script	当元素元素已被拖动到有效拖放区域时运行的脚本。
ondragleave	script	当元素离开有效拖放目标时运行的脚本。
ondragover	script	当元素在有效拖放目标上正在被拖动时运行的脚本。
ondragstart	script	在拖动操作开端运行的脚本。
ondrop	script	当被拖元素正在被拖放时运行的脚本。
onmousedown	script	当元素上按下鼠标按钮时触发。
onmousemove	script	当鼠标指针移动到元素上时触发。
onmouseout	script	当鼠标指针移出元素时触发。
onmouseover	script	当鼠标指针移动到元素上时触发。
onmouseup	script	当在元素上释放鼠标按钮时触发。
onmousewheel	script	当鼠标滚轮正在被滚动时运行的脚本。
onscroll	script	当元素滚动条被滚动时运行的脚本。

Media 事件

由媒介（比如视频、图像和音频）触发的事件（适用于所有 HTML 元素，但常见于媒介元素中，比如 <audio>、<embed>、<img>、<object> 以及 <video>）：

属性	值	描述
onabort	script	在退出时运行的脚本。
oncanplay	script	当文件就绪可以开始播放时运行的脚本（缓冲已足够开始时）。
oncanplaythrough	script	当媒介能够无需因缓冲而停止即可播放至结尾时运行的脚本。
ondurationchange	script	当媒介长度改变时运行的脚本。
		当发生故障并且文件突然不可用时运行的脚本

		（比如连接意外断开时）。
onended	script	当媒介已到达结尾时运行的脚本（可发送类似“感谢观看”之类的消息）。
onerror	script	当在文件加载期间发生错误时运行的脚本。
onloadeddata	script	当媒介数据已加载时运行的脚本。
onloadedmetadata	script	当元数据（比如分辨率和时长）被加载时运行的脚本。
onloadstart	script	在文件开始加载且未实际加载任何数据前运行的脚本。
onpause	script	当媒介被用户或程序暂停时运行的脚本。
onplay	script	当媒介已就绪可以开始播放时运行的脚本。
onplaying	script	当媒介已开始播放时运行的脚本。
onprogress	script	当浏览器正在获取媒介数据时运行的脚本。
onratechange	script	每当回放速率改变时运行的脚本（比如当用户切换到慢动作或快进模式）。
onreadystatechange	script	每当就绪状态改变时运行的脚本（就绪状态监测媒介数据的状态）。
onseeked	script	当 seeking 属性设置为 false（指示定位已结束）时运行的脚本。
onseeking	script	当 seeking 属性设置为 true（指示定位是活动的）时运行的脚本。
onstalled	script	在浏览器不论何种原因未能取回媒介数据时运行的脚本。
onsuspend	script	在媒介数据完全加载之前不论何种原因终止取回媒介数据时运行的脚本。
ontimeupdate	script	当播放位置改变时（比如当用户快进到媒介中一个不同的位置时）运行的脚本。
onvolumechange	script	每当音量改变时（包括将音量设置为静音）时运行的脚本。
onwaiting	script	当媒介已停止播放但打算继续播放时（比如当媒介暂停已缓冲更多数据）运行脚本

HTML 5 视频/音频参考手册

HTML5 视频和音频的 DOM 参考手册

HTML5 DOM 为 <audio> 和 <video> 元素提供了方法、属性和事件。

这些方法、属性和事件允许您使用 JavaScript 来操作 <audio> 和 <video> 元素。

HTML5 Audio/Video 方法

方法	描述
addTextTrack()	向音频/视频添加新的文本轨道
canPlayType()	检测浏览器是否能播放指定的音频/视频类型
load()	重新加载音频/视频元素
play()	开始播放音频/视频
pause()	暂停当前播放的音频/视频

HTML5 Audio/Video 属性

属性	描述
audioTracks	返回表示可用音轨的 AudioTrackList 对象
autoplay	设置或返回是否在加载完成后随即播放音频/视频
buffered	返回表示音频/视频已缓冲部分的 TimeRanges 对象
controller	返回表示音频/视频当前媒体控制器的 MediaController 对象
controls	设置或返回音频/视频是否显示控件（比如播放/暂停等）
crossOrigin	设置或返回音频/视频的 CORS 设置
currentSrc	返回当前音频/视频的 URL
currentTime	设置或返回音频/视频中的当前播放位置（以秒计）
defaultMuted	设置或返回音频/视频默认是否静音
defaultPlaybackRate	设置或返回音频/视频的默认播放速度

duration	返回当前音频/视频的长度（以秒计）
ended	返回音频/视频的播放是否已结束
error	返回表示音频/视频错误状态的 MediaError 对象
loop	设置或返回音频/视频是否应在结束时重新播放
mediaGroup	设置或返回音频/视频所属的组合（用于连接多个音频/视频元素）
muted	设置或返回音频/视频是否静音
networkState	返回音频/视频的当前网络状态
paused	设置或返回音频/视频是否暂停
playbackRate	设置或返回音频/视频播放的速度
played	返回表示音频/视频已播放部分的 TimeRanges 对象
preload	设置或返回音频/视频是否应该在页面加载后进行加载
readyState	返回音频/视频当前的就绪状态
seekable	返回表示音频/视频可寻址部分的 TimeRanges 对象
seeking	返回用户是否正在音频/视频中进行查找
src	设置或返回音频/视频元素的当前来源
startDate	返回表示当前时间偏移的 Date 对象
textTracks	返回表示可用文本轨道的 TextTrackList 对象
videoTracks	返回表示可用视频轨道的 VideoTrackList 对象
volume	设置或返回音频/视频的音量

HTML5 Audio/Video 事件

事件	描述
abort	当音频/视频的加载已放弃时
canplay	当浏览器可以播放音频/视频时
canplaythrough	当浏览器可在不因缓冲而停顿的情况下进行播放时
durationchange	当音频/视频的时长已更改时
emptied	当目前的播放列表为空时
ended	当目前的播放列表已结束时
error	当在音频/视频加载期间发生错误时
loadeddata	当浏览器已加载音频/视频的当前帧时
loadedmetadata	当浏览器已加载音频/视频的元数据时
loadstart	当浏览器开始查找音频/视频时
pause	当音频/视频已暂停时
play	当音频/视频已开始或不再暂停时
playing	当音频/视频在已因缓冲而暂停或停止后已就绪时
progress	当浏览器正在下载音频/视频时
ratechange	当音频/视频的播放速度已更改时
seeked	当用户已移动/跳跃到音频/视频中的新位置时
seeking	当用户开始移动/跳跃到音频/视频中的新位置时
stalled	当浏览器尝试获取媒体数据，但数据不可用时
suspend	当浏览器刻意不获取媒体数据时
timeupdate	当目前的播放位置已更改时
volumechange	当音量已更改时
waiting	当视频由于需要缓冲下一帧而停止

HTML 5 Canvas 参考手册

描述

HTML5 <canvas> 标签用于绘制图像（通过脚本，通常是 JavaScript）。

不过，<canvas> 元素本身并没有绘制能力（它仅仅是图形的容器） - 您必须使用脚本来完成实际的绘图任务。

getContext() 方法可返回一个对象，该对象提供了用于在画布上绘图的方法和属性。

本手册提供完整的 getContext("2d") 对象属性和方法，可用于在画布上绘制文本、线条、矩形、圆形等等。

浏览器支持

Internet Explorer 9、Firefox、Opera、Chrome 以及 Safari 支持 <canvas> 及其属性和方法。

注释：Internet Explorer 8 以及更早的版本不支持 <canvas> 元素。

颜色、样式和阴影

属性	描述
fillStyle	设置或返回用于填充绘画的颜色、渐变或模式
strokeStyle	设置或返回用于笔触的颜色、渐变或模式
shadowColor	设置或返回用于阴影的颜色
shadowBlur	设置或返回用于阴影的模糊级别
shadowOffsetX	设置或返回阴影距形状的水平距离
shadowOffsetY	设置或返回阴影距形状的垂直距离

方法	描述
createLinearGradient()	创建线性渐变（用在画布内容上）
createPattern()	在指定的方向上重复指定的元素
createRadialGradient()	创建放射状/环形的渐变（用在画布内容上）
addColorStop()	规定渐变对象中的颜色和停止位置

线条样式

属性	描述
lineCap	设置或返回线条的结束端点样式
lineJoin	设置或返回两条线相交时，所创建的拐角类型
lineWidth	设置或返回当前的线条宽度
miterLimit	设置或返回最大斜接长度

矩形

方法	描述
rect()	创建矩形
fillRect()	绘制“被填充”的矩形
strokeRect()	绘制矩形（无填充）
clearRect()	在给定的矩形内清除指定的像素

路径

方法	描述
fill()	填充当前绘图（路径）
stroke()	绘制已定义的路径
beginPath()	起始一条路径，或重置当前路径
moveTo()	把路径移动到画布中的指定点，不创建线条
closePath()	创建从当前点回到起始点的路径
lineTo()	添加一个新点，然后在画布中创建从该点到最后指定点的线条
clip()	从原始画布剪切任意形状和尺寸的区域
quadraticCurveTo()	创建二次贝塞尔曲线
bezierCurveTo()	创建三次方贝塞尔曲线
arc()	创建弧/曲线（用于创建圆形或部分圆）
arcTo()	创建两切线之间的弧/曲线
isPointInPath()	如果指定的点位于当前路径中，则返回 true，否则返回 false

转换

方法	描述
scale()	缩放当前绘图至更大或更小
rotate()	旋转当前绘图
translate()	重新映射画布上的 (0,0) 位置
transform()	替换绘图的当前转换矩阵
setTransform()	将当前转换重置为单位矩阵。然后运行 transform()

文本

属性	描述
font	设置或返回文本内容的当前字体属性
textAlign	设置或返回文本内容的当前对齐方式
textBaseline	设置或返回在绘制文本时使用的当前文本基线

方法	描述
fillText()	在画布上绘制“被填充的”文本
strokeText()	在画布上绘制文本（无填充）
measureText()	返回包含指定文本宽度的对象

图像绘制

方法	描述
drawImage()	向画布上绘制图像、画布或视频

像素操作

属性	描述
width	返回 ImageData 对象的宽度
height	返回 ImageData 对象的高度
data	返回一个对象，其包含指定的 ImageData 对象的图像数据

方法	描述
createImageData()	创建新的、空白的 ImageData 对象
getImageData()	返回 ImageData 对象，该对象为画布上指定的矩形复制像素数据
putImageData()	把图像数据（从指定的 ImageData 对象）放回画布上

合成

属性	描述
globalAlpha	设置或返回绘图的当前 alpha 或透明值
globalCompositeOperation	设置或返回新图像如何绘制到已有的图像上

其他

方法	描述
save()	保存当前环境的状态
restore()	返回之前保存过的路径状态和属性
createEvent()	
getContext()	
toDataURL()	

HTML 元素和有效的 DTD

HTML5/HTML 4.01/XHTML 元素和有效的 DTD

下面的表格列出了所有的 HTML5/HTML 4.01/XHTML 元素，以及它们会出现在什么文档类型 (DTD) 中：

标签	HTML5	HTML 4.01 / XHTML 1.0	XHTML 1.1		
Transitional	Strict	Frameset			
---	---	---			
<a>	Yes	Yes	Yes	Yes	Yes
<abbr>	Yes	Yes	Yes	Yes	Yes
<acronym>	No	Yes	Yes	Yes	Yes
<address>	Yes	Yes	Yes	Yes	Yes
<applet>	No	Yes	No	Yes	No
<area>	Yes	Yes	Yes	Yes	No
<article>	Yes	No	No	No	No
<aside>	Yes	No	No	No	No
<audio>	Yes	No	No	No	No
<b>	Yes	Yes	Yes	Yes	Yes
<base>	Yes	Yes	Yes	Yes	Yes
<basefont>	No	Yes	No	Yes	No
<bdi>	Yes	No	No	No	No
<bdo>	Yes	Yes	Yes	Yes	No
<big>	No	Yes	Yes	Yes	Yes
<blockquote>	Yes	Yes	Yes	Yes	Yes
<body>	Yes	Yes	Yes	Yes	Yes
 	Yes	Yes	Yes	Yes	Yes
<button>	Yes	Yes	Yes	Yes	Yes
<canvas>	Yes	No	No	No	No

<caption>	Yes	Yes	Yes	Yes	Yes
<center>	No	Yes	No	Yes	No
<cite>	Yes	Yes	Yes	Yes	Yes
<code>	Yes	Yes	Yes	Yes	Yes
<col>	Yes	Yes	Yes	Yes	No
<colgroup>	Yes	Yes	Yes	Yes	No
<command>	Yes	No	No	No	No
<datalist>	Yes	No	No	No	No
<dd>	Yes	Yes	Yes	Yes	Yes
<del>	Yes	Yes	Yes	Yes	No
<details>	Yes	No	No	No	No
<dfn>	Yes	Yes	Yes	Yes	Yes
<dir>	No	Yes	No	Yes	No
<div>	Yes	Yes	Yes	Yes	Yes
<dl>	Yes	Yes	Yes	Yes	Yes
<dt>	Yes	Yes	Yes	Yes	Yes
<em>	Yes	Yes	Yes	Yes	Yes
<embed>	Yes	No	No	No	No
<fieldset>	Yes	Yes	Yes	Yes	Yes
<figcaption>	Yes	No	No	No	No
<figure>	Yes	No	No	No	No
<font>	No	Yes	No	Yes	No
<footer>	Yes	No	No	No	No
<form>	Yes	Yes	Yes	Yes	Yes
<frame>	No	No	No	Yes	No
<frameset>	No	No	No	Yes	No
<head>	Yes	Yes	Yes	Yes	Yes
<header>	Yes	No	No	No	No
<h1> to <h6>	Yes	Yes	Yes	Yes	Yes
<hr>	Yes	Yes	Yes	Yes	Yes

<html>	Yes	Yes	Yes	Yes	Yes
<i>	Yes	Yes	Yes	Yes	Yes
<iframe>	Yes	Yes	No	Yes	No
<img>	Yes	Yes	Yes	Yes	Yes
<input>	Yes	Yes	Yes	Yes	Yes
<ins>	Yes	Yes	Yes	Yes	No
<keygen>	Yes	No	No	No	No
<kbd>	Yes	Yes	Yes	Yes	Yes
<label>	Yes	Yes	Yes	Yes	Yes
<legend>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<link>	Yes	Yes	Yes	Yes	Yes
<map>	Yes	Yes	Yes	Yes	No
<mark>	Yes	No	No	No	No
<menu>	Yes	Yes	No	Yes	No
<meta>	Yes	Yes	Yes	Yes	Yes
<meter>	Yes	No	No	No	No
<nav>	Yes	No	No	No	No
<noframes>	No	Yes	No	Yes	No
<noscript>	Yes	Yes	Yes	Yes	Yes
<object>	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	Yes
<optgroup>	Yes	Yes	Yes	Yes	Yes
<option>	Yes	Yes	Yes	Yes	Yes
<output>	Yes	No	No	No	No
<p>	Yes	Yes	Yes	Yes	Yes
<param>	Yes	Yes	Yes	Yes	Yes
<pre>	Yes	Yes	Yes	Yes	Yes
<progress>	Yes	No	No	No	No

<q>	Yes	Yes	Yes	Yes	Yes
<rp>	Yes	No	No	No	No
<rt>	Yes	No	No	No	No
<ruby>	Yes	No	No	No	No
<s>	Yes	Yes	No	Yes	No
<samp>	Yes	Yes	Yes	Yes	Yes
<script>	Yes	Yes	Yes	Yes	Yes
<section>	Yes	No	No	No	No
<select>	Yes	Yes	Yes	Yes	Yes
<small>	Yes	Yes	Yes	Yes	Yes
<source>	Yes	No	No	No	No
<span>	Yes	Yes	Yes	Yes	Yes
<strike>	No	Yes	No	Yes	No
<strong>	Yes	Yes	Yes	Yes	Yes
<style>	Yes	Yes	Yes	Yes	Yes
<sub>	Yes	Yes	Yes	Yes	Yes
<summary>	Yes	No	No	No	No
<sup>	Yes	Yes	Yes	Yes	Yes
<table>	Yes	Yes	Yes	Yes	Yes
<tbody>	Yes	Yes	Yes	Yes	No
<td>	Yes	Yes	Yes	Yes	Yes
<textarea>	Yes	Yes	Yes	Yes	Yes
<tfoot>	Yes	Yes	Yes	Yes	No
<th>	Yes	Yes	Yes	Yes	Yes
<thead>	Yes	Yes	Yes	Yes	No
<time>	Yes	No	No	No	No
<title>	Yes	Yes	Yes	Yes	Yes
<tr>	Yes	Yes	Yes	Yes	Yes
<track>	Yes	No	No	No	No
<tt>	No	Yes	Yes	Yes	Yes

<u>	No	Yes	No	Yes	No
	Yes	Yes	Yes	Yes	Yes
<var>	Yes	Yes	Yes	Yes	Yes
<video>	Yes	No	No	No	No
<wbr>	Yes	No	No	No	No

HTML 颜色名

本页提供了被大多数浏览器支持的颜色名。

提示：W3C 的 HTML 4.0 标准仅支持 16 种颜色名，它们是：aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white、yellow。

如果使用其它颜色的话，就应该使用十六进制的颜色值。

颜色名列表

单击一个颜色名或者 16 进制值，就可以查看与不同文字颜色搭配的背景颜色。

颜色名	十六进制颜色值	颜色
AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	
Aqua	#00FFFF	
Aquamarine	#7FFFD4	
Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	
Chocolate	#D2691E	
Coral	#FF7F50	
CornflowerBlue	#6495ED	

Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
Darkorange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray	#696969
DodgerBlue	#1E90FF
Feldspar	#D19275
FireBrick	#B22222
FloralWhite	#FFFAF0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC

GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateBlue	#8470FF
LightSlateGray	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0

Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE

PaleVioletRed	#D87093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8
Tomato	#FF6347
Turquoise	#40E0D0
Violet	#EE82EE

VioletRed	#D02090
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00
YellowGreen	#9ACD32

HTML 字符集

HTML 字符集

如需正确地显示 HTML 页面，浏览器必须知道使用何种字符集。

万维网早期使用的字符集是 ASCII。ASCII 支持 0-9 的数字，大写和小写英文字母表，以及一些特殊字符。

完整的 ASCII 参考手册

由于很多国家使用的字符并不属于 ASCII，现代浏览器的默认字符集是 ISO-8859-1。

完整的 ISO-8859-1 参考手册

如果网页使用不同于 ISO-8859-1 的字符集，就应该在 <meta> 标签进行指定。

ISO 字符集

ISO 字符集是国际标准组织 (ISO) 针对不同的字母表/语言定义的标准字符集。

下面列出了世界各地使用的不同字符集：

字符集	描述	使用范围
ISO-8859-1	Latin alphabet part 1	北美、西欧、拉丁美洲、加勒比海、加拿大、非洲
ISO-8859-2	Latin alphabet part 2	东欧
ISO-8859-3	Latin alphabet part 3	SE Europe、世界语、其他杂项
ISO-8859-4	Latin alphabet part 4	斯堪的纳维亚/波罗的海（以及其他没有包括在 ISO-8859-1 中的部分）
ISO-8859-5	Latin/Cyrillic part 5	使用古代斯拉夫语字母表的语言，比如保加利亚语、白俄罗斯文、俄罗斯语、马其顿语
ISO-8859-6	Latin/Arabic part 6	使用阿拉伯字母的语言
ISO-8859-7	Latin/Greek part 7	现代希腊语，以及有希腊语衍生的数学符号
ISO-8859-8	Latin/Hebrew part 8	使用希伯来语的语言
ISO-8859-9	Latin 5 part 9	土耳其语
ISO-8859-10	Latin 6	拉普兰语、日耳曼语、爱斯基摩北欧语
ISO-8859-15	Latin 9 (aka Latin 0)	与 ISO 8859-1 类似，欧元符号和其他一些字符取代了一些较少使用的符号
ISO-2022-JP	Latin/Japanese part 1	日本语
ISO-2022-JP-2	Latin/Japanese part 2	日本语
ISO-2022-KR	Latin/Korean part 1	韩语

Unicode 标准

由于上面列出的字符集都有容量限制，而且不兼容多语言环境，Unicode 联盟开发了 Unicode 标准。

Unicode 标准涵盖了世界上的所有字符、标点和符号。

不论是何种平台、程序或语言，Unicode 都能够进行文本数据的处理、存储和交换。

Unicode 联盟

Unicode 联盟开发了 Unicode 标准。他们的目标是用标准的 Unicode 转换格式 (UTF) 来取代现有的字符集。

Unicode 标准已经获得了成功，在 XML、Java、ECMAScript (JavaScript)、LDAP、CORBA 3.0、WML 中，Unicode 已经得到了实现。在许多操作系统以及所有的现代浏览器中，Unicode 同样得到了支持。

Unicode 联盟与领导性的标准发展组织进行合作，比如 ISO、W3C 以及 ECMA。

Unicode 可以被不同的字符集兼容。最常用的编码方式是 UTF-8 和 UTF-16：

UTF-8

UTF8 中的字符可以是 1-4 个字节长。UTF-8 可以表示 Unicode 标准中的任意字符。UTF-8 向后兼容 ASCII。UTF-8 是网页和电子邮件的首选编码。

UTF-16

16 比特的 Unicode 转换格式是一种 Unicode 可变字符编码，能够对全部 Unicode 指令表进行编码。UTF-16 主要被用于操作系统和环境中，比如微软的 Windows 2000/XP/2003/Vista/CE 以及 Java 和 .NET 字节代码环境。

提示：最前面的 256 个 Unicode 字符集字符对应于 256 个 ISO-8859-1 字符。

提示：所有 HTML 4 处理器均已支持 UTF-8，而所有 XHTML 和 XML 处理器支持 UTF-8 和 UTF-16。

HTML ASCII 参考手册

HTML 和 **XHTML** 用标准的 **7** 比特 **ASCII** 代码在网络上传输数据。

7 比特 **ASCII** 代码可提供 **128** 个不同的字符值。

7 比特 可显示的 ASCII 代码

结果	描述	实体编号
space		
!	exclamation mark	!
"	quotation mark	"
#	number sign	#
$	dollar sign	$
%	percent sign	%
&	ampersand	&
'	apostrophe	'
(	left parenthesis	(
)	right parenthesis	)
*	asterisk	*
+	plus sign	+
,	comma	,
-	hyphen	-
.	period	.
/	slash	/
0	digit 0	0
1	digit 1	1
2	digit 2	2
3	digit 3	3
4	digit 4	4
5	digit 5	5

6	digit 6	6
7	digit 7	7
8	digit 8	8
9	digit 9	9
:	colon	:
;	semicolon	;
<	less-than	<
=	equals-to	=
>	greater-than	>
?	question mark	?
@	at sign	@
A	uppercase A	A
B	uppercase B	B
C	uppercase C	C
D	uppercase D	D
E	uppercase E	E
F	uppercase F	F
G	uppercase G	G
H	uppercase H	H
I	uppercase I	I
J	uppercase J	J
K	uppercase K	K
L	uppercase L	L
M	uppercase M	M
N	uppercase N	N
O	uppercase O	O
P	uppercase P	P
Q	uppercase Q	Q
R	uppercase R	R

S	uppercase S	S
T	uppercase T	T
U	uppercase U	U
V	uppercase V	V
W	uppercase W	W
X	uppercase X	X
Y	uppercase Y	Y
Z	uppercase Z	Z
[	left square bracket	[
\	backslash	\
]	right square bracket	]
^	caret	^
_	underscore	_
`	grave accent	`
a	lowercase a	a
b	lowercase b	b
c	lowercase c	c
d	lowercase d	d
e	lowercase e	e
f	lowercase f	f
g	lowercase g	g
h	lowercase h	h
i	lowercase i	i
j	lowercase j	j
k	lowercase k	k
l	lowercase l	l
m	lowercase m	m
n	lowercase n	n
o	lowercase o	o
p	lowercase p	p

q	lowercase q	q
r	lowercase r	r
s	lowercase s	s
t	lowercase t	t
u	lowercase u	u
v	lowercase v	v
w	lowercase w	w
x	lowercase x	x
y	lowercase y	y
z	lowercase z	z
{	left curly brace	{
		vertical bar		
}	right curly brace	}
~	tilde	~

7 比特 设备控制 ASCII代码

ASCII设备控制代码最初被设计为用来控制诸如打印机和磁带驱动器之类的硬件设备。在HTML文档中这些代码不会起任何作用。

结果	描述	实体编号
NUL	null character	&#00;
SOH	start of header	&#01;
STX	start of text	&#02;
ETX	end of text	&#03;
EOT	end of transmission	&#04;
ENQ	enquiry	&#05;
ACK	acknowledge	&#06;
BEL	bell (ring)	&#07;
BS	backspace	&#08;
HT	horizontal tab	

| LF | line feed |
 |
|---|---|---|
| VT | vertical tab | &#11; |
| FF | form feed |  |
| CR | carriage return |
 |
| SO | shift out | &#14; |
| SI | shift in | &#15; |
| DLE | data link escape | &#16; |
| DC1 | device control 1 | &#17; |
| DC2 | device control 2 | &#18; |
| DC3 | device control 3 | &#19; |
| DC4 | device control 4 | &#20; |
| NAK | negative acknowledge | &#21; |
| SYN | synchronize | &#22; |
| ETB | end transmission block | &#23; |
| CAN | cancel | &#24; |
| EM | end of medium | &#25; |
| SUB | substitute | &#26; |
| ESC | escape | &#27; |
| FS | file separator | &#28; |
| GS | group separator | &#29; |
| RS | record separator | &#30; |
| US | unit separator | &#31; |
| DEL | delete (rubout) | &#127; |

HTML ISO-8859-1 参考手册

HTML 4.01 支持 ISO 8859-1 (Latin-1) 字符集。

ISO-8859-1 的较低部分（从 1 到 127 之间的代码）是最初的 7 比特 ASCII。

ISO-8859-1 的较高部分（从 160 到 255 之间的代码）全都有实体名称。

这些符号中的大多数都可以在不进行实体引用的情况下使用，但是实体名称或实体编号为那些不容易通过键盘键入的符号提供了表达的方法。

注释：实体名称对大小写敏感。

带有实体名称的 ASCII 实体

结果	描述	实体名称	实体编号
"	quotation mark	"	"
'	apostrophe	'	'
&	ampersand	&	&
<	less-than	<	<
>	greater-than	>	>

ISO 8859-1 符号实体

结果	描述	实体名称	实体编号
	non-breaking space		
?	inverted exclamation mark	¡	¡
¢	cent	¢	¢
£	pound	£	£
¤	currency	¤	¤
¥	yen	¥	¥
|	broken vertical bar	¦	¦
§	section	§	§
¨	spacing diaeresis	¨	¨
?	copyright	©	©

?	copyright	©	©
a	feminine ordinal indicator	ª	ª
?	angle quotation mark (left)	«	«
?	negation	¬	¬
-	soft hyphen		
?	registered trademark	®	®
¯	spacing macron	¯	¯
°	degree	°	°
±	plus-or-minus	±	±
2	superscript 2	^{2}	²
3	superscript 3	^{3}	³
′	spacing acute	´	´
μ	micro	µ	µ
?	paragraph	¶	¶
·	middle dot	·	·
?	spacing cedilla	¸	¸
1	superscript 1	^{1}	¹
o	masculine ordinal indicator	º	º
?	angle quotation mark (right)	»	»
?	fraction 1/4	¼	¼
?	fraction 1/2	½	½
?	fraction 3/4	¾	¾
?	inverted question mark	¿	¿
×	multiplication	×	×
÷	division	÷	÷

ISO 8859-1 字符实体

结果	描述	实体名称	实体编号
à	capital a, grave accent	À	À

á	capital a, acute accent	Á	Á
?	capital a, circumflex accent	Â	Â
?	capital a, tilde	Ã	Ã
?	capital a, umlaut mark	Ä	Ä
?	capital a, ring	Å	Å
?	capital ae	Æ	Æ
?	capital c, cedilla	Ç	Ç
è	capital e, grave accent	È	È
é	capital e, acute accent	É	É
ê	capital e, circumflex accent	Ê	Ê
?	capital e, umlaut mark	Ë	Ë
ì	capital i, grave accent	Ì	Ì
í	capital i, acute accent	Í	Í
?	capital i, circumflex accent	Î	Î
?	capital i, umlaut mark	Ï	Ï
D	capital eth, Icelandic	Ð	Ð
?	capital n, tilde	Ñ	Ñ
ò	capital o, grave accent	Ò	Ò
ó	capital o, acute accent	Ó	Ó
?	capital o, circumflex accent	Ô	Ô
?	capital o, tilde	Õ	Õ
?	capital o, umlaut mark	Ö	Ö
?	capital o, slash	Ø	Ø
ù	capital u, grave accent	Ù	Ù
ú	capital u, acute accent	Ú	Ú
?	capital u, circumflex accent	Û	Û
ü	capital u, umlaut mark	Ü	Ü
Y	capital y, acute accent	Ý	Ý
T	capital THORN, Icelandic	Þ	Þ
?	small sharp s, German	ß	ß

à	small a, grave accent	à	à
á	small a, acute accent	á	á
a	small a, circumflex accent	â	â
?	small a, tilde	ã	ã
?	small a, umlaut mark	ä	ä
?	small a, ring	å	å
?	small ae	æ	æ
?	small c, cedilla	ç	ç
è	small e, grave accent	è	è
é	small e, acute accent	é	é
ê	small e, circumflex accent	ê	ê
?	small e, umlaut mark	ë	ë
ì	small i, grave accent	ì	ì
í	small i, acute accent	í	í
?	small i, circumflex accent	î	î
?	small i, umlaut mark	ï	ï
e	small eth, Icelandic	ð	ð
?	small n, tilde	ñ	ñ
ò	small o, grave accent	ò	ò
ó	small o, acute accent	ó	ó
?	small o, circumflex accent	ô	ô
?	small o, tilde	õ	õ
?	small o, umlaut mark	ö	ö
?	small o, slash	ø	ø
ù	small u, grave accent	ù	ù
ú	small u, acute accent	ú	ú
?	small u, circumflex accent	û	û
ü	small u, umlaut mark	ü	ü
y	small y, acute accent	ý	ý
t	small thorn, Icelandic	þ	þ

?	small y, umlaut mark	ÿ	ÿ

HTML 4.01 符号实体

本字符实体参考手册包括了数学符号、希腊字符、各种箭头记号、科技符号以及形状。

注释：实体名称对大小写敏感。

HTML 支持的数学符号

结果	描述	实体名称	实体编号
?	for all	∀	∀
?	part	∂	∂
?	exists	&exists;	∃
?	empty	∅	∅
?	nabla	∇	∇
∈	isin	∈	∈
?	notin	∉	∉
?	ni	∋	∋
∏	prod	∏	∏
∑	sum	∑	∑
?	minus	−	−
?	lowast	∗	∗
√	square root	√	√
∝	proportional to	∝	∝
∞	infinity	∞	∞
∠	angle	∠	∠
∧	and	∧	∧
∨	or	∨	∨
∩	cap	∩	∩
∪	cup	∪	∪
∫	integral	∫	∫

∴	therefore	∴	∴
～	simular to	∼	∼
?	approximately equal	≅	≅
≈	almost equal	≈	≈
≠	not equal	≠	≠
≡	equivalent	≡	≡
≤	less or equal	≤	≤
≥	greater or equal	≥	≥
?	subset of	⊂	⊂
?	superset of	⊃	⊃
?	not subset of	⊄	⊄
?	subset or equal	⊆	⊆
?	superset or equal	⊇	⊇
⊕	circled plus	⊕	⊕
?	cirled times	⊗	⊗
⊥	perpendicular	⊥	⊥
?	dot operator	⋅	⋅

HTML 支持的希腊字母

结果	描述	实体名称	实体编号
Α	Alpha	Α	Α
Β	Beta	Β	Β
Γ	Gamma	Γ	Γ
Δ	Delta	Δ	Δ
Ε	Epsilon	Ε	Ε
Ζ	Zeta	Ζ	Ζ
Η	Eta	Η	Η
Θ	Theta	Θ	Θ
Ι	Iota	Ι	Ι

Κ	Kappa	Κ	Κ
Λ	Lambda	Λ	Λ
Μ	Mu	Μ	Μ
Ν	Nu	Ν	Ν
Ξ	Xi	Ξ	Ξ
Ο	Omicron	Ο	Ο
Π	Pi	Π	Π
Ρ	Rho	Ρ	Ρ
Sigmaf	undefined		
Σ	Sigma	Σ	Σ
Τ	Tau	Τ	Τ
Υ	Upsilon	Υ	Υ
Φ	Phi	Φ	Φ
Χ	Chi	Χ	Χ
Ψ	Psi	Ψ	Ψ
Ω	Omega	Ω	Ω
α	alpha	α	α
β	beta	β	β
γ	gamma	γ	γ
δ	delta	δ	δ
ε	epsilon	ε	ε
ζ	zeta	ζ	ζ
η	eta	η	η
θ	theta	θ	θ
ι	iota	ι	ι
κ	kappa	κ	κ
λ	lambda	λ	Λ
μ	mu	μ	μ
ν	nu	ν	Ν
ξ	xi	ξ	ξ

ο	omicron	ο	ο
π	pi	π	π
ρ	rho	ρ	ρ
?	sigmaf	ς	ς
σ	sigma	σ	σ
τ	tau	τ	τ
υ	upsilon	υ	υ
φ	phi	φ	φ
χ	chi	χ	χ
ψ	psi	ψ	ψ
ω	omega	ω	ω
?	theta symbol	ϑ	ϑ
?	upsilon symbol	ϒ	ϒ
?	pi symbol	ϖ	ϖ

HTML 支持的其他实体

结果	描述	实体名称	实体编号
?	capital ligature OE	Œ	Œ
?	small ligature oe	œ	œ
?	capital S with caron	Š	Š
?	small S with caron	š	š
?	capital Y with diaeres	Ÿ	Ÿ
?	f with hook	ƒ	ƒ
?	modifier letter circumflex accent	ˆ	ˆ
?	small tilde	˜	˜
?	en space		
?	em space		
?	thin space		
?	zero width non-joiner		‌

?	zero width joiner		‍
?	left-to-right mark		‎
?	right-to-left mark		‏
–	en dash	–	–
—	em dash	—	—
‘	left single quotation mark	‘	‘
’	right single quotation mark	’	’
?	single low-9 quotation mark	‚	‚
“	left double quotation mark	“	“
”	right double quotation mark	”	”
?	double low-9 quotation mark	„	„
?	dagger	†	†
?	double dagger	‡	‡
?	bullet	•	•
…	horizontal ellipsis	…	…
‰	per mille	‰	‰
′	minutes	′	′
″	seconds	″	″
?	single left angle quotation	‹	‹
?	single right angle quotation	›	›
‾	overline	‾	‾
€	euro	€	€
?	trademark	™	™
←	left arrow	←	←
↑	up arrow	↑	↑
→	right arrow	→	→
↓	down arrow	↓	↓
?	left right arrow	↔	↔
?	carriage return arrow	↵	↵
?	left ceiling	⌈	⌈

?	right ceiling	⌉	⌉
?	left floor	⌊	⌊
?	right floor	⌋	⌋
?	lozenge	◊	◊
?	spade	♠	♠
?	club	♣	♣
?	heart	♥	♥
?	diamond	♦	♦

HTML URL 编码

下面是用 URL 编码形式表示的 ASCII 字符（十六进制格式）。

十六进制格式用于在浏览器和插件中显示非标准的字母和字符。

URL 编码 - 从 %00 到 %8f

ASCII Value	URL-encode	ASCII Value	URL-encode	ASCII Value	URL-encode
?	%00	0	%30	`	%60
	%01	1	%31	a	%61
	%02	2	%32	b	%62
	%03	3	%33	c	%63
	%04	4	%34	d	%64
	%05	5	%35	e	%65
	%06	6	%36	f	%66
	%07	7	%37	g	%67
backspace	%08	8	%38	h	%68
tab	%09	9	%39	i	%69
linefeed	%0a	:	%3a	j	%6a
	%0b	;	%3b	k	%6b
	%0c	<	%3c	l	%6c
c return	%0d	=	%3d	m	%6d
	%0e	>	%3e	n	%6e
	%0f	?	%3f	o	%6f
	%10	@	%40	p	%70
	%11	A	%41	q	%71
	%12	B	%42	r	%72
	%13	C	%43	s	%73
	%14	D	%44	t	%74

	%15	E	%45	u	%75
	%16	F	%46	v	%76
	%17	G	%47	w	%77
	%18	H	%48	x	%78
	%19	I	%49	y	%79
	%1a	J	%4a	z	%7a
	%1b	K	%4b	{	%7b
	%1c	L	%4c	|	%7c
	%1d	M	%4d	}	%7d
	%1e	N	%4e	~	%7e
	%1f	O	%4f		%7f
space	%20	P	%50	€	%80
!	%21	Q	%51		%81
"	%22	R	%52	?	%82
#	%23	S	%53	?	%83
$	%24	T	%54	?	%84
%	%25	U	%55	…	%85
&	%26	V	%56	?	%86
'	%27	W	%57	?	%87
(	%28	X	%58	?	%88
)	%29	Y	%59	‰	%89
*	%2a	Z	%5a	?	%8a
+	%2b	[	%5b	?	%8b
,	%2c	\	%5c	?	%8c
-	%2d	]	%5d		%8d
.	%2e	^	%5e	?	%8e
/	%2f	_	%5f		%8f

URL 编码 - 从 %90 到 %ff

ASCII Value	URL-encode	ASCII Value	URL-encode	ASCII Value	URL-encode
	%90	à	%c0	e	%f0
‘	%91	á	%c1	?	%f1
’	%92	?	%c2	ò	%f2
“	%93	?	%c3	ó	%f3
”	%94	?	%c4	?	%f4
?	%95	?	%c5	?	%f5
–	%96	?	%c6	?	%f6
—	%97	?	%c7	÷	%f7
?	%98	è	%c8	?	%f8
?	%99	é	%c9	ù	%f9
?	%9a	ê	%ca	ú	%fa
?	%9b	?	%cb	?	%fb
?	%9c	ì	%cc	ü	%fc
	%9d	í	%cd	y	%fd
?	%9e	?	%ce	t	%fe
?	%9f	?	%cf	?	%ff
	%a0	D	%d0		
?	%a1	?	%d1		
¢	%a2	ò	%d2		
£	%a3	ó	%d3		
	%a4	?	%d4		
¥	%a5	?	%d5		
|	%a6	?	%d6		
§	%a7		%d7		
¨	%a8	?	%d8		
?	%a9	ù	%d9		
a	%aa	ú	%da		
?	%ab	?	%db		

?	%ac	ü	%dc
¯	%ad	Y	%dd
?	%ae	T	%de
¯	%af	?	%df
°	%b0	à	%e0
±	%b1	á	%e1
2	%b2	a	%e2
3	%b3	?	%e3
′	%b4	?	%e4
µ	%b5	?	%e5
?	%b6	?	%e6
·	%b7	?	%e7
?	%b8	è	%e8
1	%b9	é	%e9
o	%ba	ê	%ea
?	%bb	?	%eb
?	%bc	ì	%ec
?	%bd	í	%ed
?	%be	?	%ee
?	%bf	?	%ef

HTML 语言代码参考手册

ISO 语言代码

HTML 的 lang 属性可用于网页或部分网页的语言。这对搜索引擎和浏览器是有帮助的。

根据 W3C 推荐标准，您应该通过 <html> 标签中的 lang 属性对每张页面中的主要语言进行声明，比如：

```
<html lang="en">
...
</html>
```

在 XHTML 中，采用如下方式在 <html> 标签中对语言进行声明：

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
...
</html>
```

ISO 639-1 语言代码

ISO 639-1 为各种语言定义了缩略词。您可以在 HTML 和 XHTML 中的 lang 和 xml:lang 属性中使用它们。

Language	ISO Code
Abkhazian	ab
Afar	aa
Afrikaans	af
Albanian	sq
Amharic	am
Arabic	ar
Armenian	hy
Assamese	as
Aymara	ay

Azerbaijani	az
Bashkir	ba
Basque	eu
Bengali (Bangla)	bn
Bhutani	dz
Bihari	bh
Bislama	bi
Breton	br
Bulgarian	bg
Burmese	my
Byelorussian (Belarusian)	be
Cambodian	km
Catalan	ca
Cherokee	
Chewa	
Chinese (Simplified)	zh
Chinese (Traditional)	zh
Corsican	co
Croatian	hr
Czech	cs
Danish	da
Divehi	
Dutch	nl
Edo	
English	en
Esperanto	eo
Estonian	et
Faeroese	fo
Farsi	fa
Fiji	fj

Finnish	fi
Flemish	
French	fr
Frisian	fy
Fulfulde	
Galician	gl
Gaelic (Scottish)	gd
Gaelic (Manx)	gv
Georgian	ka
German	de
Greek	el
Greenlandic	kl
Guarani	gn
Gujarati	gu
Hausa	ha
Hawaiian	
Hebrew	he, iw
Hindi	hi
Hungarian	hu
Ibibio	
Icelandic	is
Igbo	
Indonesian	id, in
Interlingua	ia
Interlingue	ie
Inuktitut	iu
Inupiak	ik
Irish	ga
Italian	it
Japanese	ja

Javanese	jv
Kannada	kn
Kanuri	
Kashmiri	ks
Kazakh	kk
Kinyarwanda (Ruanda)	rw
Kirghiz	ky
Kirundi (Rundi)	rn
Konkani	
Korean	ko
Kurdish	ku
Laothian	lo
Latin	la
Latvian (Lettish)	lv
Limburgish (Limburger)	li
Lingala	ln
Lithuanian	lt
Macedonian	mk
Malagasy	mg
Malay	ms
Malayalam	ml
Maltese	mt
Maori	mi
Marathi	mr
Moldavian	mo
Mongolian	mn
Nauru	na
Nepali	ne
Norwegian	no

Occitan	oc
Oriya	or
Oromo (Afan, Galla)	om
Papiamentu	
Pashto (Pushto)	ps
Polish	pl
Portuguese	pt
Punjabi	pa
Quechua	qu
Rhaeto-Romance	rm
Romanian	ro
Russian	ru
Sami (Lappish)	
Samoan	sm
Sangro	sg
Sanskrit	sa
Serbian	sr
Serbo-Croatian	sh
Sesotho	st
Setswana	tn
Shona	sn
Sindhi	sd
Sinhalese	si
Siswati	ss
Slovak	sk
Slovenian	sl
Somali	so
Spanish	es
Sundanese	su
Swahili (Kiswahili)	sw

Swedish	sv
Syriac	
Tagalog	tl
Tajik	tg
Tamazight	
Tamil	ta
Tatar	tt
Telugu	te
Thai	th
Tibetan	bo
Tigrinya	ti
Tonga	to
Tsonga	ts
Turkish	tr
Turkmen	tk
Twi	tw
Uighur	ug
Ukrainian	uk
Urdu	ur
Uzbek	uz
Venda	
Vietnamese	vi
Volapuk	vo
Welsh	cy
Wolof	wo
Xhosa	xh
Yi	
Yiddish	yi, ji
Yoruba	yo
Zulu	zu

HTTP 状态消息

当浏览器从 **web** 服务器请求服务时，可能会发生错误。

从而有可能会返回下面的一系列状态消息：

1xx: 信息

消息:	描述:
100 Continue	服务器仅接收到部分请求，但是一旦服务器并没有拒绝该请求，客户端应该继续发送其余的请求。
101 Switching Protocols	服务器转换协议：服务器将遵从客户的请求转换到另外一种协议。

2xx: 成功

消息:	描述:
200 OK	请求成功（其后是对GET和POST请求的应答文档。）
201 Created	请求被创建完成，同时新的资源被创建。
202 Accepted	供处理的请求已被接受，但是处理未完成。
203 Non-authoritative Information	文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝。
204 No Content	没有新文档。浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的。
205 Reset Content	没有新文档。但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容。
206 Partial Content	客户发送了一个带有Range头的GET请求，服务器完成了它。

3xx: 重定向

消息:	描述:
300 Multiple Choices	多重选择。链接列表。用户可以选择某链接到达目的地。最多允许五个地址。
301 Moved Permanently	所请求的页面已经转移至新的url。
302 Found	所请求的页面已经临时转移至新的url。
303 See Other	所请求的页面可在别的url下被找到。
304 Not Modified	未按预期修改文档。客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。
305 Use Proxy	客户请求的文档应该通过Location头所指明的代理服务器提取。
306 *Unused*	此代码被用于前一版本。目前已不再使用，但是代码依然被保留。
307 Temporary Redirect	被请求的页面已经临时移至新的url。

4xx: 客户端错误

消息:	描述:
400 Bad Request	服务器未能理解请求。
401 Unauthorized	被请求的页面需要用户名和密码。
402 Payment Required	此代码尚无法使用。
403 Forbidden	对被请求页面的访问被禁止。
404 Not Found	服务器无法找到被请求的页面。
405 Method Not Allowed	请求中指定的方法不被允许。
406 Not Acceptable	服务器生成的响应无法被客户端所接受。
407 Proxy Authentication Required	用户必须首先使用代理服务器进行验证，这样请求才会被处理。
408 Request Timeout	请求超出了服务器的等待时间。
409 Conflict	由于冲突，请求无法被完成。
410 Gone	被请求的页面不可用。
411 Length Required	"Content-Length" 未被定义。如果无此内容，服务器不会接受请求。
412 Precondition Failed	请求中的前提条件被服务器评估为失败。
413 Request Entity Too Large	由于所请求的实体的太大，服务器不会接受请求。
414 Request-url Too Long	由于url太长，服务器不会接受请求。当post请求被转换为带有很长的查询信息的get请求时，就会发生这种情况。
415 Unsupported Media Type	由于媒介类型不被支持，服务器不会接受请求。
416	服务器不能满足客户在请求中指定的Range头。
417 Expectation Failed	

5xx: 服务器错误

消息:	描述:
500 Internal Server Error	请求未完成。服务器遇到不可预知的情况。
501 Not Implemented	请求未完成。服务器不支持所请求的功能。
502 Bad Gateway	请求未完成。服务器从上游服务器收到一个无效的响应。
503 Service Unavailable	请求未完成。服务器临时过载或当机。
504 Gateway Timeout	网关超时。
505 HTTP Version Not Supported	服务器不支持请求中指明的HTTP协议版本。

HTTP 方法：GET 对比 POST

两种最常用的 **HTTP** 方法是：**GET** 和 **POST**。

什么是 HTTP？

超文本传输协议（HTTP）的设计目的是保证客户机与服务器之间的通信。

HTTP 的工作方式是客户机与服务器之间的请求-应答协议。

web 浏览器可能是客户端，而计算机上的网络应用程序也可能作为服务器端。

举例：客户端（浏览器）向服务器提交 HTTP 请求；服务器向客户端返回响应。响应包含关于请求的状态信息以及可能被请求的内容。

两种 HTTP 请求方法：GET 和 POST

在客户机和服务器之间进行请求-响应时，两种最常被用到的方法是：GET 和 POST。

- *GET* - 从指定的资源请求数据。
- *POST* - 向指定的资源提交要被处理的数据

GET 方法

请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：

```
/test/demo_form.asp?name1=value1&name2=value2
```

有关 GET 请求的其他一些注释：

- GET 请求可被缓存
- GET 请求保留在浏览器历史记录中
- GET 请求可被收藏为书签
- GET 请求不应在处理敏感数据时使用
- GET 请求有长度限制
- GET 请求只应当用于取回数据

POST 方法

请注意，查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的：

```
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

有关 POST 请求的其他一些注释：

- POST 请求不会被缓存
- POST 请求不会保留在浏览器历史记录中
- POST 不能被收藏为书签
- POST 请求对数据长度没有要求

比较 GET 与 POST

下面的表格比较了两种 HTTP 方法：GET 和 POST。

	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。
对数据长度的限制	是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。	无限制。
对数据类型的限制	只允许 ASCII 字符。	没有限制。也允许二进制数据。
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。在发送密码或其他敏感信息时绝不要使用 GET ！	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
可见性	数据在 URL 中对所有人都是可见的。	数据不会显示在 URL 中。

其他 HTTP 请求方法

下面的表格列出了其他一些 HTTP 请求方法：

方法	描述
HEAD	与 GET 相同，但只返回 HTTP 报头，不返回文档主体。
PUT	上传指定的 URI 表示。
DELETE	删除指定资源。
OPTIONS	返回服务器支持的 HTTP 方法。
CONNECT	把请求连接转换到透明的 TCP/IP 通道。

W3School CSS教程

来源：CSS教程

整理：飞龙

CSS 样式

CSS 背景

CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果。

CSS 在这方面的能力远远在 **HTML** 之上。

背景色

可以使用 background-color 属性为元素设置背景色。这个属性接受任何合法的颜色值。

这条规则把元素的背景设置为灰色：

```
p {background-color: gray;}
```

如果您希望背景色从元素中的文本向外少有延伸，只需增加一些内边距：

```
p {background-color: gray; padding: 20px;}
```

如需查看本例的效果，可以！

可以为所有元素设置背景色，这包括 body 一直到 em 和 a 等行内元素。

background-color 不能继承，其默认值是 transparent。transparent 有“透明”之意。也就是说，如果一个元素没有指定背景色，那么背景就是透明的，这样其祖先元素的背景才能可见。

背景图像

要把图像放入背景，需要使用 background-image 属性。background-image 属性的默认值是 none，表示背景上没有放置任何图像。

如果需要设置一个背景图像，必须为这个属性设置一个 URL 值：

```
body {background-image: url(img/eg_bg_04.gif);}
```

大多数背景都应用到 body 元素，不过并不仅限于此。

下面例子为一个段落应用了一个背景，而不会对文档的其他部分应用背景：

```
p.flower {background-image: url(img/eg_bg_03.gif);}
```

您甚至可以为行内元素设置背景图像，下面的例子为一个链接设置了背景图像：

```
a.radio {background-image: url(img/eg_bg_07.gif);}
```

如需查看上述例子的效果，可以！

理论上讲，甚至可以向 textareas 和 select 等替换元素的背景应用图像，不过并不是所有用户代理都能很好地处理这种情况。

另外还要补充一点，background-image 也不能继承。事实上，所有背景属性都不能继承。

背景重复

如果需要在页面上对背景图像进行平铺，可以使用 background-repeat 属性。

属性值 repeat 导致图像在水平垂直方向上都平铺，就像以往背景图像的通常做法一样。repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复，no-repeat 则不允许图像在任何方向上平铺。

默认地，背景图像将从一个元素的左上角开始。请看下面的例子：

```
body
  {
  background-image: url(img/eg_bg_03.gif);
  background-repeat: repeat-y;
  }
```

如需查看上例的效果，可以。

背景定位

可以利用 background-position 属性改变图像在背景中的位置。

下面的例子在 body 元素中将一个背景图像居中放置：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:center;
  }
```

为 background-position 属性提供值有很多方法。首先，可以使用一些关键字：top、bottom、left、right 和 center。通常，这些关键字会成对出现，不过也不总是这样。还可以使用长度值，如 100px 或 5cm，最后也可以使用百分数值。不同类型的值对于背景图像的放置稍有差异。

关键字

图像放置关键字最容易理解，其作用如其名称所表明的。例如，top right 使图像放置在元素内边距区的右上角。

根据规范，位置关键字可以按任何顺序出现，只要保证不超过两个关键字 - 一个对应水平方向，另一个对应垂直方向。

如果只出现一个关键字，则认为另一个关键字是 center。

所以，如果希望每个段落的中部上方出现一个图像，只需声明如下：

```
p
  {
    background-image:url('bgimg.gif');
    background-repeat:no-repeat;
    background-position:top;
  }
```

下面是等价的位置关键字：

单一关键字	等价的关键字
center	center center
top	top center 或 center top
bottom	bottom center 或 center bottom
right	right center 或 center right
left	left center 或 center left

百分数值

百分数值的表现方式更为复杂。假设你希望用百分数值将图像在其元素中居中，这很容易：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:50% 50%;
  }
```

这会导致图像适当放置，其中心与其元素的中心对齐。换句话说，百分数值同时应用于元素和图像。也就是说，图像中描述为 50% 50% 的点（中心点）与元素中描述为 50% 50% 的点（中心点）对齐。

如果图像位于 0% 0%，其左上角将放在元素内边距区的左上角。如果图像位置是 100% 100%，会使图像的右下角放在右边距的右下角。

因此，如果你想把一个图像放在水平方向 2/3、垂直方向 1/3 处，可以这样声明：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:66% 33%;
  }
```

如果只提供一个百分数值，所提供的这个值将用作水平值，垂直值将假设为 50%。这一点与关键字类似。

background-position 的默认值是 0% 0%，在功能上相当于 top left。这就解释了背景图像为什么总是从元素内边距区的左上角开始平铺，除非您设置了不同的位置值。

长度值

长度值解释的是元素内边距区左上角的偏移。偏移点是图像的左上角。

比如，如果设置值为 50px 100px，图像的左上角将在元素内边距区左上角向右 50 像素、向下 100 像素的位置上：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:50px 100px;
  }
```

注意，这一点与百分数值不同，因为偏移只是从一个左上角到另一个左上角。也就是说，图像的左上角与 background-position 声明中的指定的点对齐。

背景关联

如果文档比较长，那么当文档向下滚动时，背景图像也会随之滚动。当文档滚动到超过图像的位置时，图像就会消失。

您可以通过 background-attachment 属性防止这种滚动。通过这个属性，可以声明图像相对于可视区是固定的（fixed），因此不会受到滚动的影响：

```
body
  {
  background-image:url(img/eg_bg_02.gif);
  background-repeat:no-repeat;
  background-attachment:fixed
  }
```

如需查看上例的效果，可以。

background-attachment 属性的默认值是 scroll，也就是说，在默认的情况下，背景会随文档滚动。

CSS 背景实例

设置背景颜色

设置文本的背景颜色

将图像设置为背景

将图像设置为背景 2

如何重复背景图像

如何在垂直方向重复背景图像

如何在水平方向重复背景图像

如何仅显示一次背景图像

如何放置背景图像

如何使用%来定位背景图像

如何使用像素来定位背景图像

如何设置固定的背景图像

所有背景属性在一个声明之中

CSS 背景属性

属性	描述
background	简写属性，作用是将背景属性设置在一个声明中。
background-attachment	背景图像是否固定或者随着页面的其余部分滚动。
background-color	设置元素的背景颜色。
background-image	把图像设置为背景。
background-position	设置背景图像的起始位置。
background-repeat	设置背景图像是否及如何重复。

CSS 文本

CSS 文本属性可定义文本的外观。

通过文本属性，您可以改变文本的颜色、字符间距，对齐文本，装饰文本，对文本进行缩进，等等。

缩进文本

把 Web 页面上的段落的第一行缩进，这是一种最常用的文本格式化效果。

CSS 提供了 text-indent 属性，该属性可以方便地实现文本缩进。

通过使用 text-indent 属性，所有元素的第一行都可以缩进一个给定的长度，甚至该长度可以是负值。

这个属性最常见的用途是将段落的首行缩进，下面的规则会使所有段落的首行缩进 5 em：

```
p {text-indent: 5em;}
```

注意：一般来说，可以为所有块级元素应用 text-indent，但无法将该属性应用于行内元素，图像之类的替换元素上也无法应用 text-indent 属性。不过，如果一个块级元素（比如段落）的首行中有一个图像，它会随该行的其余文本移动。

提示：如果想把一个行内元素的第一行“缩进”，可以用左内边距或外边距创造这种效果。

使用负值

text-indent 还可以设置为负值。利用这种技术，可以实现很多有趣的效果，比如“悬挂缩进”，即第一行悬挂在元素中余下部分的左边：

```
p {text-indent: -5em;}
```

不过在为 text-indent 设置负值时要当心，如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距：

```
p {text-indent: -5em; padding-left: 5em;}
```

使用百分比值

text-indent 可以使用所有长度单位，包括百分比值。

百分数要相对于缩进元素父元素的宽度。换句话说，如果将缩进值设置为 20%，所影响元素的第一行会缩进其父元素宽度的 20%。

在下例中，缩进值是父元素的 20%，即 100 个像素：

```
div {width: 500px;}
p {text-indent: 20%;}

<div>
<p>this is a paragragh</p>
</div>
```

继承

text-indent 属性可以继承，请考虑如下标记：

```
div#outer {width: 500px;}
div#inner {text-indent: 10%;}
p {width: 200px;}

<div id="outer">
<div id="inner">some text. some text. some text.
<p>this is a paragragh.</p>
</div>
</div>
```

以上标记中的段落也会缩进 50 像素，这是因为这个段落继承了 id 为 inner 的 div 元素的缩进值。

水平对齐

text-align 是一个基本的属性，它会影响一个元素中的文本行互相之间的对齐方式。它的前 3 个值相当直接，不过第 4 个和第 5 个则略有些复杂。

值 left、right 和 center 会导致元素中的文本分别左对齐、右对齐和居中。

西方语言都是从左向右读，所有 text-align 的默认值是 left。文本在左边界对齐，右边界呈锯齿状（称为“从左到右”文本）。对于希伯来语和阿拉伯语之类的的语言，text-align 则默认为 right，因为这些语言从右向左读。不出所料，center 会使每个文本行在元素中居中。

提示：将块级元素或表元素居中，要通过在这些元素上适当地设置左、右外边距来实现。

text-align:center 与 <CENTER>

您可能会认为 text-align:center 与 <CENTER> 元素的作用一样，但实际上二者大不相同。

<CENTER> 不仅影响文本，还会把整个元素居中。text-align 不会控制元素的对齐，而只影响内部内容。元素本身不会从一段移到另一端，只是其中的文本受影响。

justify

最后一个水平对齐属性是 justify。

在两端对齐文本中，文本行的左右两端都放在父元素的内边界上。然后，调整单词和字母间的间隔，使各行的长度恰好相等。您也许已经注意到了，两端对齐文本在打印领域很常见。

需要注意的是，要由用户代理（而不是 CSS）来确定两端对齐文本如何拉伸，以填满父元素左右边界之间的空间。如需了解详情，请参阅 CSS text-align 属性参考页。

字间隔

word-spacing 属性可以改变字（单词）之间的标准间隔。其默认值 normal 与设置值为 0 是一样的。

word-spacing 属性接受一个正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把它拉近：

```
p.spread {word-spacing: 30px;}
p.tight {word-spacing: -0.5em;}

<p class="spread">
This is a paragraph. The spaces between words will be increased.
</p>

<p class="tight">
This is a paragraph. The spaces between words will be decreased.
</p>
```

实例 TIY ：增加或减少单词间距（字间隔）

注释：如需深入理解 CSS 对“字”（word）的定义，请访问 CSS word-spacing 属性参考页。

字母间隔

letter-spacing 属性与 word-spacing 的区别在于，字母间隔修改的是字符或字母之间的间隔。

与 word-spacing 属性一样，letter-spacing 属性的可取值包括所有长度。默认关键字是 normal（这与 letter-spacing:0 相同）。输入的长度值会使字母之间的间隔增加或减少指定的量：

```
h1 {letter-spacing: -0.5em}
h4 {letter-spacing: 20px}

<h1>This is header 1</h1>
<h4>This is header 4</h4>
```

实例 TIY ：规定字符间距（字母间隔）

字符转换

text-transform 属性处理文本的大小写。这个属性有 4 个值：

- none
- uppercase
- lowercase
- capitalize

默认值 none 对文本不做任何改动，将使用源文档中的原有大小写。顾名思义，uppercase 和 lowercase 将文本转换为全大写和全小写字符。最后，capitalize 只对每个单词的首字母大写。

作为一个属性，text-transform 可能无关紧要，不过如果您突然决定把所有 h1 元素变为大写，这个属性就很有用。不必单独地修改所有 h1 元素的内容，只需使用 text-transform 为你完成这个修改：

```
h1 {text-transform: uppercase}
```

使用 text-transform 有两方面的好处。首先，只需写一个简单的规则来完成这个修改，而无需修改 h1 元素本身。其次，如果您以后决定将所有大小写再切换为原来的大小写，可以更容易地完成修改。

实例 TIY ：控制文本中字母的大小写

文本装饰

接下来，我们讨论 text-decoration 属性，这是一个很有意思的属性，它提供了很多非常有趣的行为。

text-decoration 有 5 个值：

- none
- underline
- overline
- line-through
- blink

不出所料，underline 会对元素加下划线，就像 HTML 中的 U 元素一样。overline 的作用恰好相反，会在文本的顶端画一个上划线。值 line-through 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 strike 元素。blink 会让文本闪烁，类似于 Netscape 支持的颇招非议的 blink 标记。

none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观，但也不总是这样。例如，链接默认地会有下划线。如果您希望去掉超链接的下划线，可以使用以下 CSS 来做到这一点：

```
a {text-decoration: none;}
```

注意：如果显式地用这样一个规则去掉链接的下划线，那么锚与正常文本之间在视觉上的唯一差别就是颜色（至少默认是这样的，不过也不能完全保证其颜色肯定有区别）。

还可以在一个规则中结合多种装饰。如果希望所有超链接既有下划线，又有上划线，则规则如下：

```
a:link a:visited {text-decoration: underline overline;}
```

不过要注意的是，如果两个不同的装饰都与同一元素匹配，胜出规则的值会完全取代另一个值。请考虑以下的规则：

```
h2.stricken {text-decoration: line-through;}
h2 {text-decoration: underline overline;}
```

对于给定的规则，所有 class 为 stricken 的 h2 元素都只有一个贯穿线装饰，而没有下划线和上划线，因为 *text-decoration* 值会替换而不是累积起来。

处理空白符

white-space 属性会影响到用户代理对源文档中的空格、换行和 tab 字符的处理。

通过使用该属性，可以影响浏览器处理字之间和文本行之间的空白符的方式。从某种程度上讲，默认的 XHTML 处理已经完成了空白符处理：它会把所有空白符合并为一个空格。所以给定以下标记，它在 Web 浏览器中显示时，各个字之间只会显示一个空格，同时忽略元素中的换行：

```
<p>This     paragraph has     many
    spaces           in it.</p>
```

可以用以下声明显式地设置这种默认行为：

```
p {white-space: normal;}
```

上面的规则告诉浏览器按照平常的做法去处理：丢掉多余的空白符。如果给定这个值，换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。

实例 TIY ：white-space: normal

值 pre

不过，如果将 white-space 设置为 pre，受这个属性影响的元素中，空白符的处理就有所不同，其行为就像 XHTML 的 pre 元素一样；空白符不会被忽略。

如果 white-space 属性的值为 pre，浏览器将会注意额外的空格，甚至回车。在这个方面，而且仅在这个方面，任何元素都可以相当于一个 pre 元素。

实例 TIY ：white-space: pre

注意：经测试，IE 7 以及更早版本的浏览器不支持该值，因此请使用非 IE 的浏览器来查看上面的实例。

值 nowrap

与之相对的值是 nowrap，它会防止元素中的文本换行，除非使用了一个 br 元素。在 CSS 中使用 nowrap 非常类似于 HTML 4 中用 <td nowrap> 将一个表单元格设置为不能换行，不过 white-space 值可以应用到任何元素。

实例 TIY ：white-space: nowrap

值 pre-wrap 和 pre-line

CSS2.1 引入了值 pre-wrap 和 pre-line，这在以前版本的 CSS 中是没有的。这些值的作用是允许创作人员更好地控制空白符处理。

如果元素的 white-space 设置为 pre-wrap，那么该元素中的文本会保留空白符序列，但是文本行会正常地换行。如果设置为这个值，源文本中的行分隔符以及生成的行分隔符也会保留。pre-line 与 pre-wrap 相反，会像正常文本中一样合并空白符序列，但保留换行符。

实例 TIY ：white-space: pre-wrap

实例 TIY ：white-space: pre-line

注意：我们在 IE7 和 FireFox2.0 浏览器中测试了上面的两个实例，但是结果是，值 pre-wrap 和 pre-line 都没有得到很好的支持。

总结

下面的表格总结了 white-space 属性的行为：

值	空白符	换行符	自动换行
pre-line	合并	保留	允许
normal	合并	忽略	允许
nowrap	合并	忽略	不允许
pre	保留	保留	不允许
pre-wrap	保留	保留	允许

文本方向

如果您阅读的是英文书籍，就会从左到右、从上到下地阅读，这就是英文的流方向。不过，并不是所有语言都如此。我们知道古汉语就是从右到左来阅读的，当然还包括希伯来语和阿拉伯语等等。CSS2 引入了一个属性来描述其方向性。

direction 属性影响块级元素中文本的书写方向、表中列布局的方向、内容水平填充其元素框的方向、以及两端对齐元素中最后一行的位置。

注释：对于行内元素，只有当 unicode-bidi 属性设置为 embed 或 bidi-override 时才会应用 direction 属性。

direction 属性有两个值：ltr 和 rtl。大多数情况下，默认值是 ltr，显示从左到右的文本。如果显示从右到左的文本，应使用值 rtl。

CSS 文本实例：

设置文本颜色

设置文本的背景颜色

规定字符间距

使用百分比设置行间距

使用像素值设置行间距

使用数值来设置行间距

对齐文本

修饰文本

缩进文本

控制文本中的字母

在元素中禁止文本折行

增加单词间距

CSS 文本属性

属性	描述
color	设置文本颜色
direction	设置文本方向。
line-height	设置行高。
letter-spacing	设置字符间距。
text-align	对齐元素中的文本。
text-decoration	向文本添加修饰。
text-indent	缩进元素中文本的首行。
text-shadow	设置文本阴影。CSS2 包含该属性，但是 CSS2.1 没有保留该属性。
text-transform	控制元素中的字母。
unicode-bidi	设置文本方向。
white-space	设置元素中空白的处理方式。
word-spacing	设置字间距。

CSS 字体

CSS 字体属性定义文本的字体系列、大小、加粗、风格（如斜体）和变形（如小型大写字母）。

CSS 字体系列

在 CSS 中，有两种不同类型的字体系列名称：

- 通用字体系列 - 拥有相似外观的字体系统组合（比如 "Serif" 或 "Monospace"）
- 特定字体系列 - 具体的字体系列（比如 "Times" 或 "Courier"）

除了各种特定的字体系列外，CSS 定义了 5 种通用字体系列：

- Serif 字体
- Sans-serif 字体
- Monospace 字体
- Cursive 字体
- Fantasy 字体

如果需要了解更多有关字体系列的知识，请阅读 CSS 字体系列。

指定字体系列

使用 font-family 属性 定义文本的字体系列。

使用通用字体系列

如果你希望文档使用一种 sans-serif 字体，但是你并不关心是哪一种字体，以下就是一个合适的声明：

```
body {font-family: sans-serif;}
```

这样用户代理就会从 sans-serif 字体系列中选择一个字体（如 Helvetica），并将其应用到 body 元素。因为有继承，这种字体选择还将应用到 body 元素中包含的所有元素，除非有一种更特定的选择器将其覆盖。

指定字体系列

除了使用通用的字体系列，您还可以通过 font-family 属性设置更具体的字体。

下面的例子为所有 h1 元素设置了 Georgia 字体：

```
h1 {font-family: Georgia;}
```

这样的规则同时会产生另外一个问题，如果用户代理上没有安装 Georgia 字体，就只能使用用户代理的默认字体来显示 h1 元素。

我们可以通过结合特定字体名和通用字体系列来解决这个问题：

```
h1 {font-family: Georgia, serif;}
```

如果读者没有安装 Georgia，但安装了 Times 字体（serif 字体系列中的一种字体），用户代理就可能对 h1 元素使用 Times。尽管 Times 与 Georgia 并不完全匹配，但至少足够接近。

因此，我们建议在所有 font-family 规则中都提供一个通用字体系列。这样就提供了一条后路，在用户代理无法提供与规则匹配的特定字体时，就可以选择一个候选字体。

如果您对字体非常熟悉，也可以为给定的元素指定一系列类似的字体。要做到这一点，需要把这些字体按照优先顺序排列，然后用逗号进行连接：

```
p {font-family: Times, TimesNR, 'New Century Schoolbook',
     Georgia, 'New York', serif;}
```

根据这个列表，用户代理会按所列的顺序查找这些字体。如果列出的所有字体都不可用，就会简单地选择一种可用的 serif 字体。

使用引号

您也许已经注意到了，上面的例子中使用了单引号。只有当字体名中有一个或多个空格（比如 New York），或者如果字体名包括 # 或 $ 之类的符号，才需要在 font-family 声明中加引号。

单引号或双引号都可以接受。但是，如果把一个 font-family 属性放在 HTML 的 style 属性中，则需要使用该属性本身未使用的那种引号：

```
<p style="font-family: Times, TimesNR, 'New Century Schoolbook', Ge
 'New York', serif;">...</p>
```

字体风格

font-style 属性最常用于规定斜体文本。

该属性有三个值：

- normal - 文本正常显示
- italic - 文本斜体显示
- oblique - 文本倾斜显示

实例

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

italic 和 oblique 的区别

font-style 非常简单：用于在 normal 文本、italic 文本和 oblique 文本之间选择。唯一有点复杂的是明确 italic 文本和 oblique 文本之间的差别。

斜体（italic）是一种简单的字体风格，对每个字母的结构有一些小改动，来反映变化的外观。与此不同，倾斜（oblique）文本则是正常竖直文本的一个倾斜版本。

通常情况下，italic 和 oblique 文本在 web 浏览器中看上去完全一样。

字体变形

font-variant 属性可以设定小型大写字母。

小型大写字母不是一般的大写字母，也不是小写字母，这种字母采用不同大小的大写字母。

实例

```
p {font-variant:small-caps;}
```

字体加粗

font-weight 属性设置文本的粗细。

使用 bold 关键字可以将文本设置为粗体。

关键字 100 ~ 900 为字体指定了 9 级加粗度。如果一个字体内置了这些加粗级别，那么这些数字就直接映射到预定义的级别，100 对应最细的字体变形，900 对应最粗的字体变形。数字 400 等价于 normal，而 700 等价于 bold。

如果将元素的加粗设置为 bolder，浏览器会设置比所继承值更粗的一个字体加粗。与此相反，关键词 lighter 会导致浏览器将加粗度下移而不是上移。

实例

```
p.normal {font-weight:normal;}
p.thick {font-weight:bold;}
p.thicker {font-weight:900;}
```

字体大小

font-size 属性设置文本的大小。

有能力管理文本的大小在 web 设计领域很重要。但是，您不应当通过调整文本大小使段落看上去像标题，或者使标题看上去像段落。

请始终使用正确的 HTML 标题，比如使用 <h1> - <h6> 来标记标题，使用 <p> 来标记段落。

font-size 值可以是绝对或相对值。

绝对值：

- 将文本设置为指定的大小
- 不允许用户在所有浏览器中改变文本大小（不利于可用性）
- 绝对大小在确定了输出的物理尺寸时很有用

相对大小：

- 相对于周围的元素来设置大小
- 允许用户在浏览器改变文本大小

注意：如果您没有规定字体大小，普通文本（比如段落）的默认大小是 16 像素 (16px=1em)。

使用像素来设置字体大小

通过像素设置文本大小，可以对文本大小进行完全控制：

实例

```
h1 {font-size:60px;}
h2 {font-size:40px;}
p {font-size:14px;}
```

在 Firefox, Chrome, and Safari 中，可以重新调整以上例子的文本大小，但是在 Internet Explorer 中不行。

虽然可以通过浏览器的缩放工具调整文本大小，但是这实际上是对整个页面的调整，而不仅限于文本。

使用 em 来设置字体大小

如果要避免在 Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替 pixels。

W3C 推荐使用 em 尺寸单位。

1em 等于当前的字体尺寸。如果一个元素的 font-size 为 16 像素，那么对于该元素，1em 就等于 16 像素。在设置字体大小时，em 的值会相对于父元素的字体大小改变。

浏览器中默认的文本大小是 16 像素。因此 1em 的默认尺寸是 16 像素。

可以使用下面这个公式将像素转换为 em：*pixels*/16=*em*

（注：16 等于父元素的默认字体大小，假设父元素的 font-size 为 20px，那么公式需改为：*pixels*/20=*em*）

实例

```
h1 {font-size:3.75em;} /* 60px/16=3.75em */
h2 {font-size:2.5em;}  /* 40px/16=2.5em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

在上面的例子中，以 em 为单位的文本大小与前一个例子中以像素计的文本是相同的。不过，如果使用 em 单位，则可以在所有浏览器中调整文本大小。

不幸的是，在 IE 中仍存在问题。在重设文本大小时，会比正常的尺寸更大或更小。

结合使用百分比和 EM

在所有浏览器中均有效的方案是为 body 元素（父元素）以百分比设置默认的 font-size 值：

实例

```
body {font-size:100%;}
h1 {font-size:3.75em;}
h2 {font-size:2.5em;}
p {font-size:0.875em;}
```

我们的代码非常有效。在所有浏览器中，可以显示相同的文本大小，并允许所有浏览器缩放文本的大小。

CSS 字体实例：

设置文本的字体

设置字体尺寸

设置字体风格

设置字体的异体

设置字体的粗细

所有字体属性在一个声明之内

CSS 字体属性

属性	描述
font	简写属性。作用是把所有针对字体的属性设置在一个声明中。
font-family	设置字体系列。
font-size	设置字体的尺寸。
font-size-adjust	当首选字体不可用时，对替换字体进行智能缩放。（CSS2.1 已删除该属性。）
font-stretch	对字体进行水平拉伸。（CSS2.1 已删除该属性。）
font-style	设置字体风格。
font-variant	以小型大写字体或者正常字体显示文本。
font-weight	设置字体的粗细。

CSS 链接

我们能够以不同的方法为链接设置样式。

设置链接的样式

能够设置链接样式的 CSS 属性有很多种（例如 color, font-family, background 等等）。

链接的特殊性在于能够根据它们所处的状态来设置它们的样式。

链接的四种状态：

- a:link - 普通的、未被访问的链接
- a:visited - 用户已访问的链接
- a:hover - 鼠标指针位于链接的上方
- a:active - 链接被点击的时刻

实例

```
a:link {color:#FF0000;}        /* 未被访问的链接 */
a:visited {color:#00FF00;}    /* 已被访问的链接 */
a:hover {color:#FF00FF;}    /* 鼠标指针移动到链接上 */
a:active {color:#0000FF;}    /* 正在被点击的链接 */
```

当为链接的不同状态设置样式时，请按照以下次序规则：

- a:hover 必须位于 a:link 和 a:visited 之后
- a:active 必须位于 a:hover 之后

常见的链接样式

在上面的例子中，链接根据其状态改变颜色。

让我们看看其他几种常见的设置链接样式的方法：

文本修饰

text-decoration 属性大多用于去掉链接中的下划线：

实例

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

背景色

background-color 属性规定链接的背景色：

实例

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

更多实例

向链接添加不同的样式

高级 - 创建链接框

CSS 列表

CSS 列表属性允许你放置、改变列表项标志，或者将图像作为列表项标志。

CSS 列表

从某种意义上讲，不是描述性的文本的任何内容都可以认为是列表。人口普查、太阳系、家谱、参观菜单，甚至你的所有朋友都可以表示为一个列表或者是列表的列表。

由于列表如此多样，这使得列表相当重要，所以说，CSS 中列表样式不太丰富确实是一大憾事。

列表类型

要影响列表的样式，最简单（同时支持最充分）的办法就是改变其标志类型。

例如，在一个无序列表中，列表项的标志 (marker) 是出现在各列表项旁边的圆点。在有序列表中，标志可能是字母、数字或另外某种计数体系中的一个符号。

要修改用于列表项的标志类型，可以使用属性 list-style-type：

```
ul {list-style-type : square}
```

上面的声明把无序列表中的列表项标志设置为方块。

列表项图像

有时，常规的标志是不够的。你可能想对各标志使用一个图像，这可以利用 list-style-image 属性做到：

```
ul li {list-style-image : url(xxx.gif)}
```

只需要简单地使用一个 url() 值，就可以使用图像作为标志。

列表标志位置

CSS2.1 可以确定标志出现在列表项内容之外还是内容内部。这是利用 list-style-position 完成的。

简写列表样式

为简单起见，可以将以上 3 个列表样式属性合并为一个方便的属性：list-style，就像这样：

```
li {list-style : url(example.gif) square inside}
```

list-style 的值可以按任何顺序列出，而且这些值都可以忽略。只要提供了一个值，其它的就会填入其默认值。

CSS 列表实例：

在无序列表中的不同类型的列表标记

在有序列表中不同类型的列表项标记

所有的列表样式类型

将图像作为列表项标记

放置列表标记

在一个声明中定义所有的列表属性

CSS 列表属性(list)

属性	描述
list-style	简写属性。用于把所有用于列表的属性设置于一个声明中。
list-style-image	将图象设置为列表项标志。
list-style-position	设置列表中列表项标志的位置。
list-style-type	设置列表项标志的类型。
marker-offset	

CSS 表格

CSS 表格属性可以帮助您极大地改善表格的外观。

表格边框

如需在 CSS 中设置表格边框，请使用 border 属性。

下面的例子为 table、th 以及 td 设置了蓝色边框：

```
table, th, td
  {
  border: 1px solid blue;
  }
```

请注意，上例中的表格具有双线条边框。这是由于 table、th 以及 td 元素都有独立的边框。

如果需要把表格显示为单线条边框，请使用 border-collapse 属性。

折叠边框

border-collapse 属性设置是否将表格边框折叠为单一边框：

```
table
  {
  border-collapse:collapse;
  }

table,th, td
  {
  border: 1px solid black;
  }
```

表格宽度和高度

通过 width 和 height 属性定义表格的宽度和高度。

下面的例子将表格宽度设置为 100%，同时将 th 元素的高度设置为 50px：

```
table
  {
  width:100%;
  }

th
  {
  height:50px;
  }
```

表格文本对齐

text-align 和 vertical-align 属性设置表格中文本的对齐方式。

text-align 属性设置水平对齐方式，比如左对齐、右对齐或者居中：

```
td
  {
  text-align:right;
  }
```

vertical-align 属性设置垂直对齐方式，比如顶部对齐、底部对齐或居中对齐：

```
td
  {
  height:50px;
  vertical-align:bottom;
  }
```

表格内边距

如需控制表格中内容与边框的距离，请为 td 和 th 元素设置 padding 属性：

```
td
  {
  padding:15px;
  }
```

表格颜色

下面的例子设置边框的颜色，以及 th 元素的文本和背景颜色：

```
table, td, th
  {
  border:1px solid green;
  }

th
  {
  background-color:green;
  color:white;
  }
```

CSS Table 属性

属性	描述
border-collapse	设置是否把表格边框合并为单一的边框。
border-spacing	设置分隔单元格边框的距离。
caption-side	设置表格标题的位置。
empty-cells	设置是否显示表格中的空单元格。
table-layout	设置显示单元、行和列的算法。

亲自试一试 - 更多实例

制作一个漂亮的表格

显示表格中的空单元

设置表格边框之间的空白

设置表格标题的位置

CSS 轮廓

轮廓（outline）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

CSS outline 属性规定元素轮廓的样式、颜色和宽度。

轮廓（Outline） 实例：

在元素周围画线

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
border:red solid thin;
outline:#00ff00 dotted thick;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时, Internet E
&lt;/body&gt;
&lt;/html&gt;
```

设置轮廓的颜色

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
border:red solid thin;
outline-style:dotted;
outline-color:#00ff00;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时, Internet E
&lt;/body&gt;
&lt;/html&gt;
```

设置轮廓的样式

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
border: red solid thin;
}
p.dotted {outline-style: dotted}
p.dashed {outline-style: dashed}
p.solid {outline-style: solid}
p.double {outline-style: double}
p.groove {outline-style: groove}
p.ridge {outline-style: ridge}
p.inset {outline-style: inset}
p.outset {outline-style: outset}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p class="dotted"&gt;A dotted outline&lt;/p&gt;
&lt;p class="dashed"&gt;A dashed outline&lt;/p&gt;
&lt;p class="solid"&gt;A solid outline&lt;/p&gt;
&lt;p class="double"&gt;A double outline&lt;/p&gt;
&lt;p class="groove"&gt;A groove outline&lt;/p&gt;
&lt;p class="ridge"&gt;A ridge outline&lt;/p&gt;
&lt;p class="inset"&gt;An inset outline&lt;/p&gt;
&lt;p class="outset"&gt;An outset outline&lt;/p&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时, Internet E
&lt;/body&gt;
&lt;/html&gt;
```

设置轮廓的宽度

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p.one
{
border:red solid thin;
outline-style:solid;
outline-width:thin;
}
p.two
{
border:red solid thin;
outline-style:dotted;
outline-width:3px;
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p class="one"&gt;This is some text in a paragraph.&lt;/p&gt;
&lt;p class="two"&gt;This is some text in a paragraph.&lt;/p&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时，Internet E
&lt;/body&gt;
&lt;/html&gt;
```

CSS 边框属性

"CSS" 列中的数字指示哪个 CSS 版本定义了该属性。

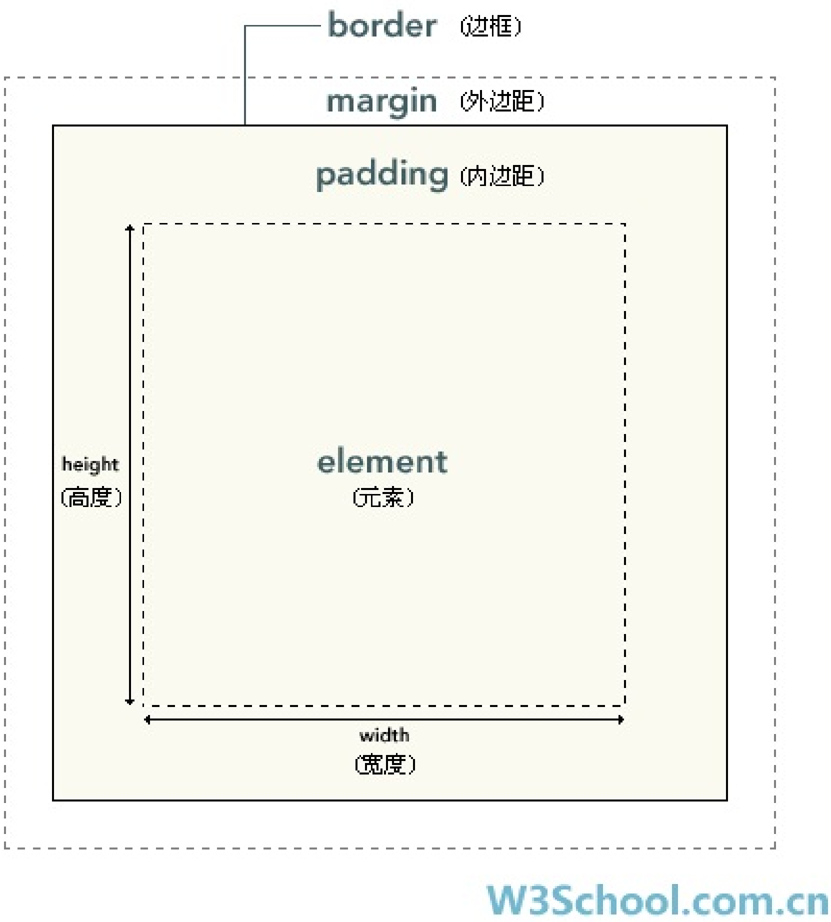
属性	描述	CSS
outline	在一个声明中设置所有的轮廓属性。	2
outline-color	设置轮廓的颜色。	2
outline-style	设置轮廓的样式。	2
outline-width	设置轮廓的宽度。	2

CSS 框模型

CSS 框模型概述

CSS 框模型 (Box Model) 规定了元素框处理元素内容、内边距、边框 和 外边距 的方式。

CSS 框模型概述



元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

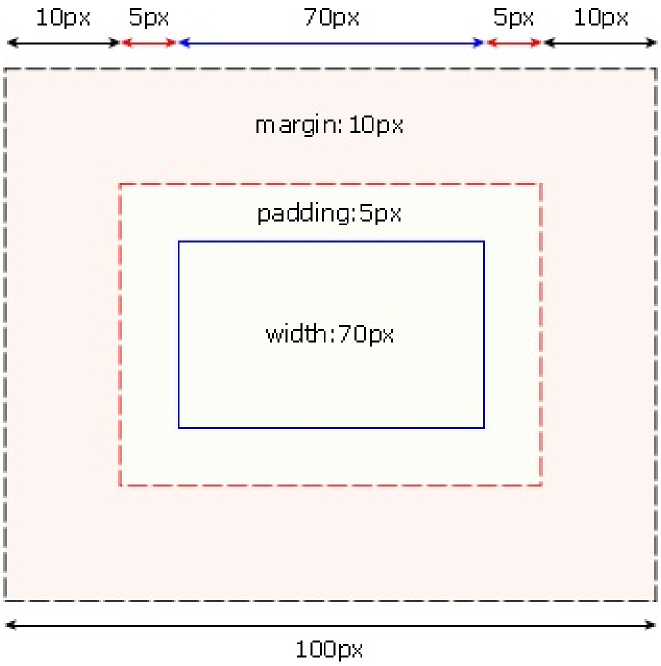
提示：背景应用于由内容和内边距、边框组成的区域。

内边距、边框和外边距都是可选的，默认值是零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 margin 和 padding 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置：

```
* {
  margin: 0;
  padding: 0;
}
```

在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。

假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，请看下图：




```
#box {
  width: 70px;
  margin: 10px;
  padding: 5px;
}
```

提示：内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。

提示：外边距可以是负值，而且在很多情况下都要使用负值的外边距。

浏览器兼容性

一旦为页面设置了恰当的 DTD，大多数浏览器都会按照上面的图示来呈现内容。然而 IE 5 和 6 的呈现却是不正确的。根据 W3C 的规范，元素内容占据的空间是由 width 属性设置的，而内容周围的 padding 和 border 值是另外计算的。不幸的是，IE5.X 和 6 在怪异模式中使用自己的非标准模型。这些浏览器的 width 属性不是内容的宽度，而是内容、内边距和边框的宽度的总和。

虽然有方法解决这个问题。但是目前最好的解决方案是回避这个问题。也就是，不要给元素添加具有指定宽度的内边距，而是尝试将内边距或外边距添加到元素的父元素和子元素。

术语翻译

- element : 元素。
- padding : 内边距，也有资料将其翻译为填充。
- border : 边框。
- margin : 外边距，也有资料将其翻译为空白或空白边。

在 w3school，我们把 padding 和 margin 统一地称为内边距和外边距。边框内的空白是内边距，边框外的空白是外边距，很容易记吧：）

CSS 内边距

元素的内边距在边框和内容区之间。控制该区域最简单的属性是 **padding** 属性。

CSS padding 属性定义元素边框与元素内容之间的空白区域。

CSS padding 属性

CSS padding 属性定义元素的内边距。padding 属性接受长度值或百分比值，但不允许使用负值。

例如，如果您希望所有 h1 元素的各边都有 10 像素的内边距，只需要这样：

```
h1 {padding: 10px;}
```

您还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
h1 {padding: 10px 0.25em 2ex 20%;}
```

单边内边距属性

也通过使用下面四个单独的属性，分别设置上、右、下、左内边距：

- padding-top
- padding-right
- padding-bottom
- padding-left

您也许已经想到了，下面的规则实现的效果与上面的简写规则是完全相同的：

```
h1 {
  padding-top: 10px;
  padding-right: 0.25em;
  padding-bottom: 2ex;
  padding-left: 20%;
  }
```

内边距的百分比数值

前面提到过，可以为元素的内边距设置百分数值。百分数值是相对于其父元素的 width 计算的，这一点与外边距一样。所以，如果父元素的 width 改变，它们也会改变。

下面这条规则把段落的内边距设置为父元素 width 的 10%：

```
p {padding: 10%;}
```

例如：如果一个段落的父元素是 div 元素，那么它的内边距要根据 div 的 width 计算。

```
<div style="width: 200px;">
<p>This paragragh is contained within a DIV that has a width of 200
</div>
```

注意：上下内边距与左右内边距一致；即上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度。

CSS 内边距实例：

所有内边距属性在一个声明中

设置下内边距 1

设置下内边距 2

设置左内边距 1

设置左内边距 2

设置右内边距 1

设置右内边距 2

设置上内边距 1

设置上内边距 2

CSS 内边距属性

属性	描述
padding	简写属性。作用是在一个声明中设置元素的所内边距属性。
padding-bottom	设置元素的下内边距。
padding-left	设置元素的左内边距。
padding-right	设置元素的右内边距。
padding-top	设置元素的上内边距。

CSS 边框

元素的边框 (border) 是围绕元素内容和内边距的一条或多条线。

CSS border 属性允许你规定元素边框的样式、宽度和颜色。

CSS 边框

在 HTML 中，我们使用表格来创建文本周围的边框，但是通过使用 CSS 边框属性，我们可以创建出效果出色的边框，并且可以应用于任何元素。

元素外边距内就是元素的的边框 (border)。元素的边框就是围绕元素内容和内边据的一条或多条线。

每个边框有 3 个方面：宽度、样式，以及颜色。在下面的篇幅，我们会为您详细讲解这三个方面。

边框与背景

CSS 规范指出，边框绘制在“元素的背景之上”。这很重要，因为有些边框是“间断的”（例如，点线边框或虚线框），元素的背景应当出现在边框的可见部分之间。

CSS2 指出背景只延伸到内边距，而不是边框。后来 CSS2.1 进行了更正：元素的背景是内容、内边距和边框区的背景。大多数浏览器都遵循 CSS2.1 定义，不过一些较老的浏览器可能会有不同的表现。

边框的样式

样式是边框最重要的一个方面，这不是因为样式控制着边框的显示（当然，样式确实控制着边框的显示），而是因为如果没有样式，将根本没有边框。

CSS 的 border-style 属性定义了 10 个不同的非 inherit 样式，包括 none。

例如，您可以为把一幅图片的边框定义为 outset，使之看上去像是“凸起按钮”：

```
a:link img {border-style: outset;}
```

定义多种样式

您可以为一个边框定义多个样式，例如：

```
p.aside {border-style: solid dotted dashed double;}
```

上面这条规则为类名为 aside 的段落定义了四种边框样式：实线上边框、点线右边框、虚线下边框和一个双线左边框。

我们又看到了这里的值采用了 top-right-bottom-left 的顺序，讨论用多个值设置不同内边距时也见过这个顺序。

定义单边样式

如果您希望为元素框的某一个边设置边框样式，而不是设置所有 4 个边的边框样式，可以使用下面的单边边框样式属性：

- border-top-style
- border-right-style
- border-bottom-style
- border-left-style

因此这两种方法是等价的：

```
p {border-style: solid solid solid none;}
p {border-style: solid; border-left-style: none;}
```

注意：如果要使用第二种方法，必须把单边属性放在简写属性之后。因为如果把单边属性放在 border-style 之前，简写属性的值就会覆盖单边值 none。

边框的宽度

您可以通过 border-width 属性为边框指定宽度。

为边框指定宽度有两种方法：可以指定长度值，比如 2px 或 0.1em；或者使用 3 个关键字之一，它们分别是 thin 、medium（默认值） 和 thick。

注释：CSS 没有定义 3 个关键字的具体宽度，所以一个用户代理可能把 thin 、medium 和 thick 分别设置为等于 5px、3px 和 2px，而另一个用户代理则分别设置为 3px、2px 和 1px。

所以，我们可以这样设置边框的宽度：

```
p {border-style: solid; border-width: 5px;}
```

或者：

```
p {border-style: solid; border-width: thick;}
```

定义单边宽度

您可以按照 top-right-bottom-left 的顺序设置元素的各边边框：

```
p {border-style: solid; border-width: 15px 5px 15px 5px;}
```

上面的例子也可以简写为（这样写法称为值复制）：

```
p {border-style: solid; border-width: 15px 5px;}
```

您也可以通过下列属性分别设置边框各边的宽度：

- border-top-width
- border-right-width
- border-bottom-width
- border-left-width

因此，下面的规则与上面的例子是等价的：

```
p {
  border-style: solid;
  border-top-width: 15px;
  border-right-width: 5px;
  border-bottom-width: 15px;
  border-left-width: 5px;
  }
```

没有边框

在前面的例子中，您已经看到，如果希望显示某种边框，就必须设置边框样式，比如 solid 或 outset。

那么如果把 border-style 设置为 none 会出现什么情况：

```
p {border-style: none; border-width: 50px;}
```

尽管边框的宽度是 50px，但是边框样式设置为 none。在这种情况下，不仅边框的样式没有了，其宽度也会变成 0。边框消失了，为什么呢？

这是因为如果边框样式为 none，即边框根本不存在，那么边框就不可能有宽度，因此边框宽度自动设置为 0，而不论您原先定义的是什么。

记住这一点非常重要。事实上，忘记声明边框样式是一个常犯的错误。根据以下规则，所有 h1 元素都不会有任何边框，更不用说 20 像素宽了：

```
h1 {border-width: 20px;}
```

由于 border-style 的默认值是 none，如果没有声明样式，就相当于 border-style: none。因此，如果您希望边框出现，就必须声明一个边框样式。

边框的颜色

设置边框颜色非常简单。CSS 使用一个简单的 border-color 属性，它一次可以接受最多 4 个颜色值。

可以使用任何类型的颜色值，例如可以是命名颜色，也可以是十六进制和 RGB 值：

```
p {
  border-style: solid;
  border-color: blue rgb(25%,35%,45%) #909090 red;
  }
```

如果颜色值小于 4 个，值复制就会起作用。例如下面的规则声明了段落的上下边框是蓝色，左右边框是红色：

```
p {
  border-style: solid;
  border-color: blue red;
  }
```

注释：默认的边框颜色是元素本身的前景色。如果没有为边框声明颜色，它将与元素的文本颜色相同。另一方面，如果元素没有任何文本，假设它是一个表格，其中只包含图像，那么该表的边框颜色就是其父元素的文本颜色（因为 color 可以继承）。这个父元素很可能是 body、div 或另一个 table。

定义单边颜色

还有一些单边边框颜色属性。它们的原理与单边样式和宽度属性相同：

- border-top-color
- border-right-color
- border-bottom-color
- border-left-color

要为 h1 元素指定实线黑色边框，而右边框为实线红色，可以这样指定：

```
h1 {
  border-style: solid;
  border-color: black;
  border-right-color: red;
  }
```

透明边框

我们刚才讲过，如果边框没有样式，就没有宽度。不过有些情况下您可能希望创建一个不可见的边框。

CSS2 引入了边框颜色值 transparent。这个值用于创建有宽度的不可见边框。请看下面的例子：

```
<a href="#">AAA</a>
<a href="#">BBB</a>
<a href="#">CCC</a>
```

我们为上面的链接定义了如下样式：

```
a:link, a:visited {
  border-style: solid;
  border-width: 5px;
  border-color: transparent;
  }
a:hover {border-color: gray;}
```

如需查看以上样式的效果，请点击：。

从某种意义上说，利用 transparent，使用边框就像是额外的内边距一样；此外还有一个好处，就是能在你需要的时候使其可见。这种透明边框相当于内边距，因为元素的背景会延伸到边框区域（如果有可见背景的话）。

重要事项：在 IE7 之前，IE/WIN 没有提供对 transparent 的支持。在以前的版本，IE 会根据元素的 color 值来设置边框颜色。

CSS 边框实例：

所有边框属性在一个声明之中

设置四边框样式

设置每一边的不同边框

所有边框宽度属性在一个声明之中

设置四个边框的颜色

所有下边框属性在一个声明中

设置下边框的颜色

设置下边框的样式

设置下边框的宽度

所有左边框属性在一个声明之中

设置左边框的颜色

设置左边框的样式

设置左边框的宽度

所有右边框属性在一个声明之中

设置右边框的颜色

设置右边框的样式

设置右边框的宽度

所有上边框属性在一个声明之中

设置上边框的颜色

设置上边框的样式

设置上边框的宽度

CSS 边框属性

属性	描述
border	简写属性，用于把针对四个边的属性设置在一个声明。
border-style	用于设置元素所有边框的样式，或者单独地为各边设置边框样式。
border-width	简写属性，用于为元素的所有边框设置宽度，或者单独地为各边边框设置宽度。
border-color	简写属性，设置元素的所有边框中可见部分的颜色，或为 4 个边分别设置颜色。
border-bottom	简写属性，用于把下边框的所有属性设置到一个声明中。
border-bottom-color	设置元素的下边框的颜色。
border-bottom-style	设置元素的下边框的样式。
border-bottom-width	设置元素的下边框的宽度。
border-left	简写属性，用于把左边框的所有属性设置到一个声明中。
border-left-color	设置元素的左边框的颜色。

border-left-style	设置元素的左边框的样式。
border-left-width	设置元素的左边框的宽度。
border-right	简写属性，用于把右边框的所有属性设置到一个声明中。
border-right-color	设置元素的右边框的颜色。
border-right-style	设置元素的右边框的样式。
border-right-width	设置元素的右边框的宽度。
border-top	简写属性，用于把上边框的所有属性设置到一个声明中。
border-top-color	设置元素的上边框的颜色。
border-top-style	设置元素的上边框的样式。
border-top-width	设置元素的上边框的宽度。

CSS 外边距

围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。

设置外边距的最简单的方法就是使用 **margin** 属性，这个属性接受任何长度单位、百分数值甚至负值。

CSS margin 属性

设置外边距的最简单的方法就是使用 margin 属性。

margin 属性接受任何长度单位，可以是像素、英寸、毫米或 em。

margin 可以设置为 auto。更常见的做法是为外边距设置长度值。下面的声明在 h1 元素的各个边上设置了 1/4 英寸宽的空白：

```
h1 {margin : 0.25in;}
```

下面的例子为 h1 元素的四个边分别定义了不同的外边距，所使用的长度单位是像素 (px)：

```
h1 {margin : 10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距 (top) 开始围着元素顺时针旋转的：

```
margin: top right bottom left
```

另外，还可以为 margin 设置一个百分比数值：

```
p {margin : 10%;}
```

百分数是相对于父元素的 width 计算的。上面这个例子为 p 元素设置的外边距是其父元素的 width 的 10%。

margin 的默认值是 0，所以如果没有为 margin 声明一个值，就不会出现外边距。但是，在实际中，浏览器对许多元素已经提供了预定的样式，外边距也不例外。例如，在支持 CSS 的浏览器中，外边距会在每个段落元素的上面和下面生成“空行”。因此，如果没有为 p 元素声明外边距，浏览器可能会自己应用一个外边距。当然，只要你特别作了声明，就会覆盖默认样式。

值复制

还记得吗？我们曾经在前两节中提到过值复制。下面我们为您讲解如何使用值复制。

有时，我们会输入一些重复的值：

```
p {margin: 0.5em 1em 0.5em 1em;}
```

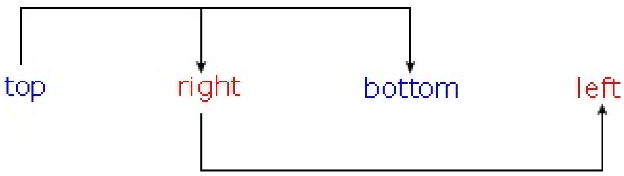
通过值复制，您可以不必重复地键入这对数字。上面的规则与下面的规则是等价的：

```
p {margin: 0.5em 1em;}
```

这两个值可以取代前面 4 个值。这是如何做到的呢？CSS 定义了一些规则，允许为外边距指定少于 4 个值。规则如下：

- 如果缺少左外边距的值，则使用右外边距的值。
- 如果缺少下外边距的值，则使用上外边距的值。
- 如果缺少右外边距的值，则使用上外边距的值。

下图提供了更直观的方法来了解这一点：



换句话说，如果为外边距指定了 3 个值，则第 4 个值（即左外边距）会从第 2 个值（右外边距）复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值（下外边距）会从第 1 个值（上外边距）复制得到。最后一个情况，如果只给定一个值，那么其他 3 个外边距都由这个值（上外边距）复制得到。

利用这个简单的机制，您只需指定必要的值，而不必全部都应用 4 个值，例如：

```
h1 {margin: 0.25em 1em 0.5em;}     /* 等价于 0.25em 1em 0.5em 1em */
h2 {margin: 0.5em 1em;}          /* 等价于 0.5em 1em 0.5em 1em */
p {margin: 1px;}               /* 等价于 1px 1px 1px 1px */
```

这种办法有一个小缺点，您最后肯定会遇到这个问题。假设希望把 p 元素的上外边距和左外边距设置为 20 像素，下外边距和右外边距设置为 30 像素。在这种情况下，必须写作：

```
p {margin: 20px 30px 30px 20px;}
```

这样才能得到您想要的结果。遗憾的是，在这种情况下，所需值的个数没有办法更少了。

再来看另外一个例子。如果希望除了左外边距以外所有其他外边距都是 auto（左外边距是 20px）：

```
p {margin: auto auto auto 20px;}
```

同样的，这样才能得到你想要的效果。问题在于，键入这些 auto 有些麻烦。如果您只是希望控制元素单边上的外边距，请使用单边外边距属性。

单边外边距属性

您可以使用单边外边距属性为元素单边上的外边距设置值。假设您希望把 p 元素的左外边距设置为 20px。不必使用 margin（需要键入很多 auto），而是可以采用以下方法：

```
p {margin-left: 20px;}
```

您可以使用下列任何一个属性来只设置相应上的外边距，而不会直接影响所有其他外边距：

- margin-top
- margin-right
- margin-bottom
- margin-left

一个规则中可以使用多个这种单边属性，例如：

```
h2 {
  margin-top: 20px;
  margin-right: 30px;
  margin-bottom: 30px;
  margin-left: 20px;
  }
```

当然，对于这种情况，使用 margin 可能更容易一些：

```
p {margin: 20px 30px 30px 20px;}
```

不论使用单边属性还是使用 margin，得到的结果都一样。一般来说，如果希望为多个边设置外边距，使用 margin 会更容易一些。不过，从文档显示的角度看，实际上使用哪种方法都不重要，所以应该选择对自己来说更容易的一种方法。

提示和注释

提示：Netscape 和 IE 对 body 标签定义的默认边距（margin）值是 8px。而 Opera 不是这样。相反地，Opera 将内部填充（padding）的默认值定义为 8px，因此如果希望对整个网站的边缘部分进行调整，并将之正确显示于 Opera 中，那么必须对 body 的 padding 进行自定义。

CSS 外边距实例：

设置文本的左外边距

设置文本的右外边距

设置文本的上外边距

设置文本的下外边距

所有的外边距属性在一个声明中。

CSS 外边距属性

属性	描述
margin	简写属性。在一个声明中设置所有外边距属性。
margin-bottom	设置元素的下外边距。
margin-left	设置元素的左外边距。
margin-right	设置元素的右外边距。
margin-top	设置元素的上外边距。

CSS 外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

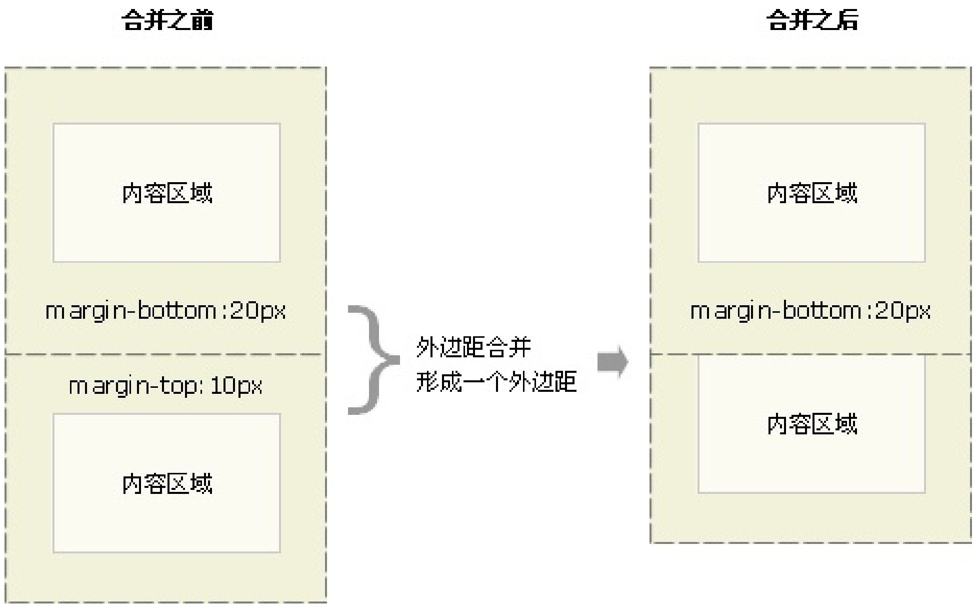
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

外边距合并

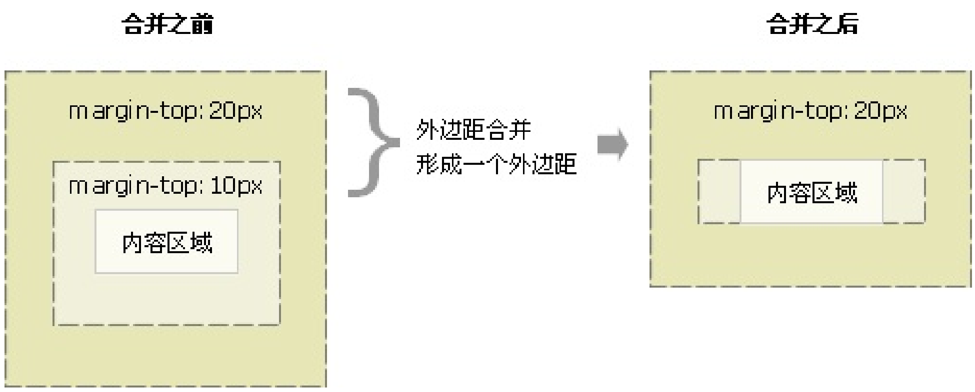
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

简单地说，外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。请看下图：

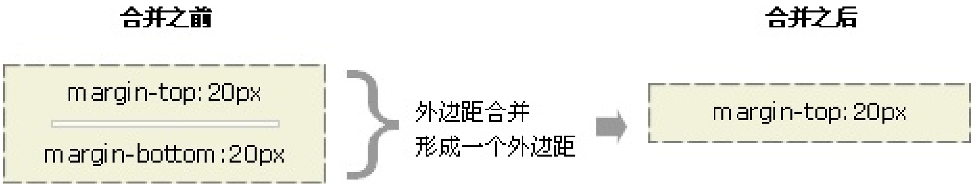


当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：

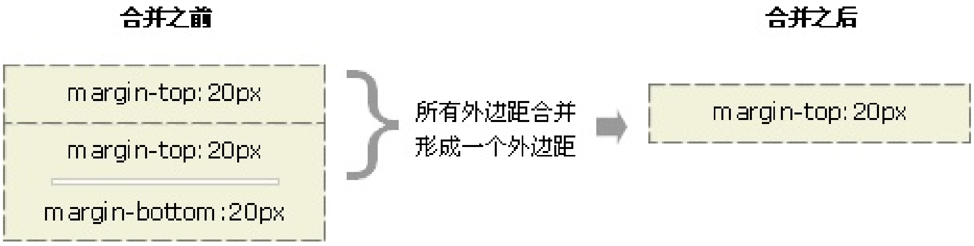


尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并：

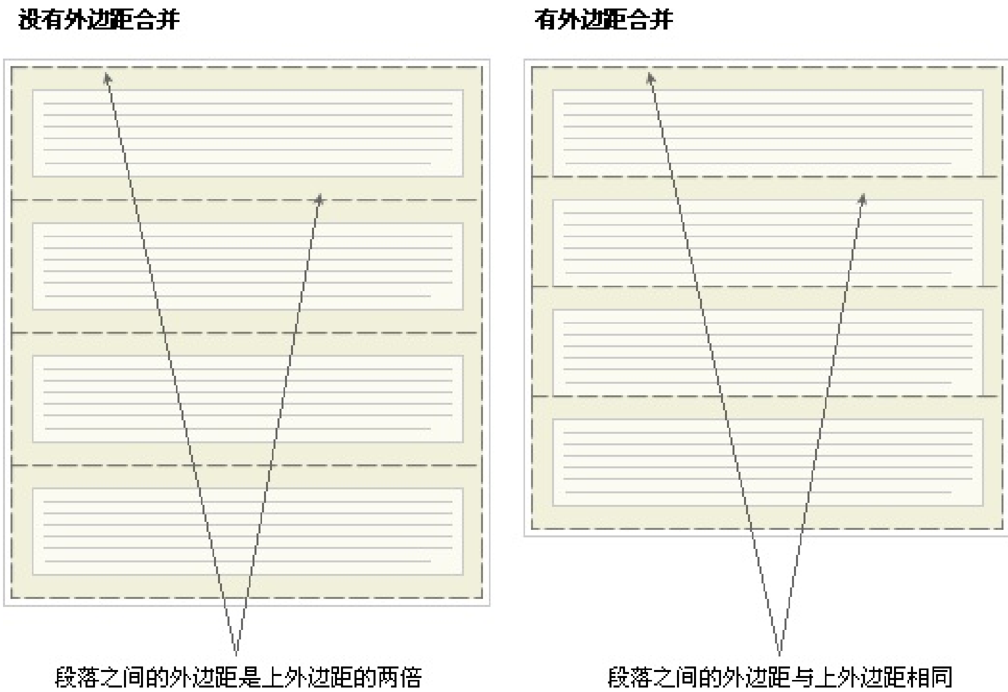


如果这个外边距遇到另一个元素的外边距，它还会发生合并：



这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

外边距合并初看上去可能有点奇怪，但是实际上，它是有意义的。以由几个段落组成的典型文本页面为例。第一个段落上面的空间等于段落的上外边距。如果没有外边距合并，后续所有段落之间的外边距都将是相邻上外边距和下外边距的和。这意味着段落之间的空间是页面顶部的两倍。如果发生外边距合并，段落之间的上外边距和下外边距就合并在一起，这样各处的距离就一致了。



注释：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

CSS 定位

CSS 定位 (Positioning)

CSS 定位 (Positioning) 属性允许你对元素进行定位。

CSS 定位和浮动

CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。显然，这个功能非常强大，也很让人吃惊。要知道，用户代理对 CSS2 中定位的支持远胜于对其它方面的支持，对此不应感到奇怪。

另一方面，CSS1 中首次提出了浮动，它以 Netscape 在 Web 发展初期增加的一个功能为基础。浮动不完全是定位，不过，它当然也不是正常流布局。我们会在后面的章节中明确浮动的含义。

一切皆为框

div、h1 或 p 元素常常被称为块级元素。这意味着这些元素显示为一块内容，即“块框”。与之相反，span 和 strong 等元素称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”。

您可以使用 display 属性改变生成的框的类型。这意味着，通过将 display 属性设置为 block，可以让行内元素（比如 <a> 元素）表现得像块级元素一样。还可以通过把 display 设置为 none，让生成的元素根本没有框。这样的话，该框及其所有内容就不再显示，不占用文档中的空间。

但是在一种情况下，即使没有进行显式定义，也会创建块级元素。这种情况发生在把一些文本添加到一个块级元素（比如 div）的开头。即使没有把这些文本定义为段落，它也会被当作段落对待：

```
<div>
some text
<p>Some more text.</p>
</div>
```

在这种情况下，这个框称为无名块框，因为它不与专门定义的元素相关联。

块级元素的文本行也会发生类似的情况。假设有一个包含三行文本的段落。每行文本形成一个无名框。无法直接对无名块或行框应用样式，因为没有可以应用样式的地方（注意，行框和行内框是两个概念）。但是，这有助于理解在屏幕上看到的所

有东西都形成某种框。

CSS 定位机制

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

除非专门指定，否则所有框都在普通流中定位。也就是说，普通流中的元素的位置由元素在 (X)HTML 中的位置决定。

块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来。

行内框在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。由一行形成的水平框称为行框（*Line Box*），行框的高度总是足以容纳它包含的所有行内框。不过，设置行高可以增加这个框的高度。

在下面的章节，我们会为您详细讲解相对定位、绝对定位和浮动。

CSS position 属性

通过使用 position 属性，我们可以选择 4 种不同类型的定位，这会影响元素框生成的方式。

position 属性值的含义：

static

relative

absolute

fixed

提示：相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

实例

定位：相对定位

定位：绝对定位

定位：固定定位

使用固定值设置图像的上边缘

使用百分比设置图像的上边缘

使用像素值设置图像的底部边缘

使用百分比设置图像的底部边缘

使用固定值设置图像的左边缘

使用百分比设置图像的左边缘

使用固定值设置图像的右边缘

使用百分比设置图像的右边缘

如何使用滚动条来显示元素内溢出的内容

如何隐藏溢出元素中溢出的内容

如何设置浏览器来自动地处理溢出

设置元素的形状

垂直排列图象

Z-index

Z-index

CSS 定位属性

CSS 定位属性允许你对元素进行定位。

属性	描述
position	把元素放置到一个静态的、相对的、绝对的、或固定的位置中。
top	定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移。
right	定义了定位元素右外边距边界与其包含块右边界之间的偏移。
bottom	定义了定位元素下外边距边界与其包含块下边界之间的偏移。
left	定义了定位元素左外边距边界与其包含块左边界之间的偏移。
overflow	设置当元素的内容溢出其区域时发生的事情。
clip	设置元素的形状。元素被剪入这个形状之中，然后显示出来。
vertical-align	设置元素的垂直对齐方式。
z-index	设置元素的堆叠顺序。

CSS 相对定位

设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，它原本所占的空间仍保留。

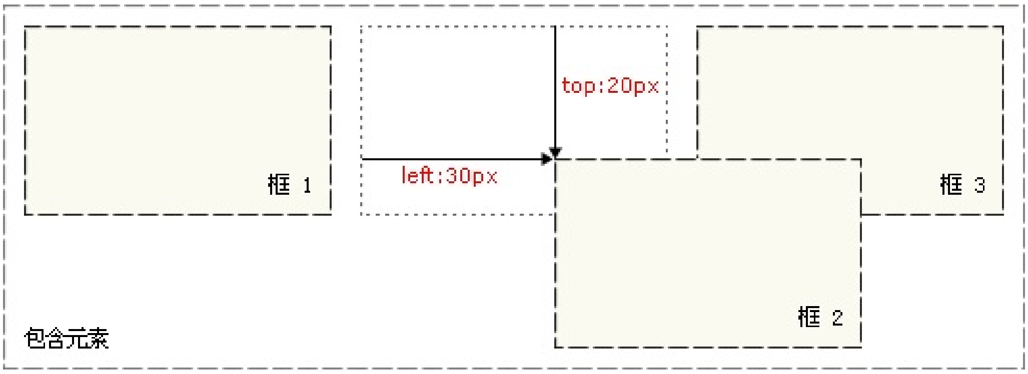
CSS 相对定位

相对定位是一个非常容易掌握的概念。如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。

如果将 top 设置为 20px，那么框将在原位置顶部下面 20 像素的地方。如果 left 设置为 30 像素，那么会在元素左边创建 30 像素的空间，也就是将元素向右移动。

```
#box_relative {
  position: relative;
  left: 30px;
  top: 20px;
}
```

如下图所示：



注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

CSS 相对定位实例

定位：相对定位

CSS 绝对定位

设置为绝对定位的元素框从文档流完全删除，并相对于其包含块定位，包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

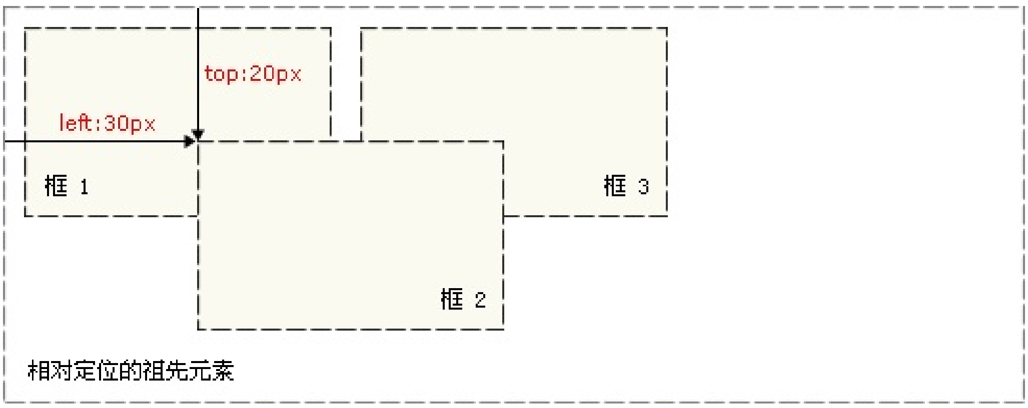
CSS 绝对定位

绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

普通流中其它元素的布局就像绝对定位的元素不存在一样：

```
#box_relative {
  position: absolute;
  left: 30px;
  top: 20px;
}
```

如下图所示：



绝对定位的元素的位置相对于最近的已定位祖先元素，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块。

对于定位的主要问题是要记住每种定位的意义。所以，现在让我们复习一下学过的知识吧：相对定位是“相对于”元素在文档中的初始位置，而绝对定位是“相对于”最近的已定位祖先元素，如果不存在已定位的祖先元素，那么“相对于”最初的包含块。

注释：根据用户代理的不同，最初的包含块可能是画布或 HTML 元素。

提示：因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 z-index 属性来控制这些框的堆放次序。

CSS 绝对定位实例

定位：绝对定位

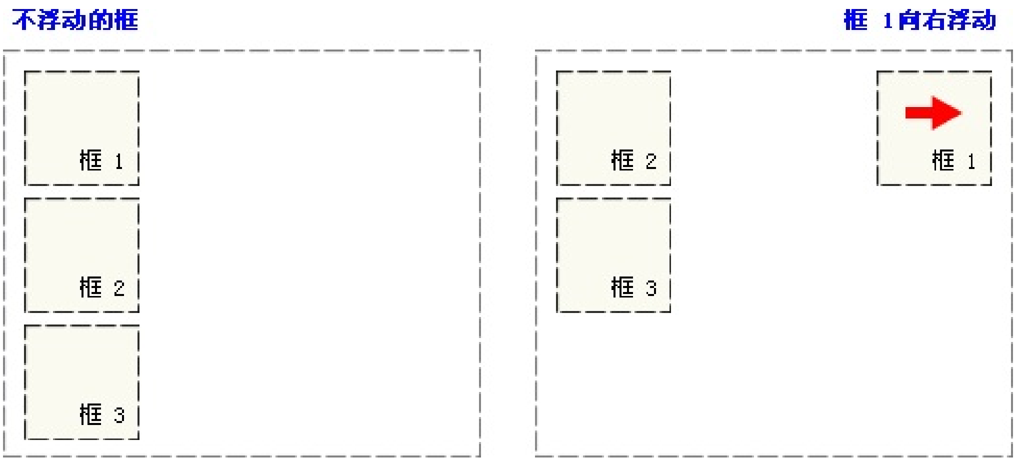
CSS 浮动

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

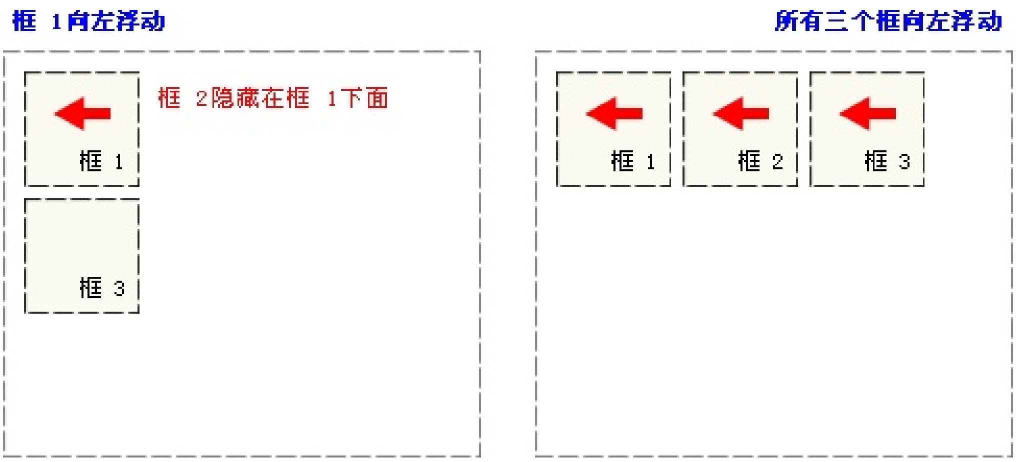
CSS 浮动

请看下图，当把框 1 向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘：

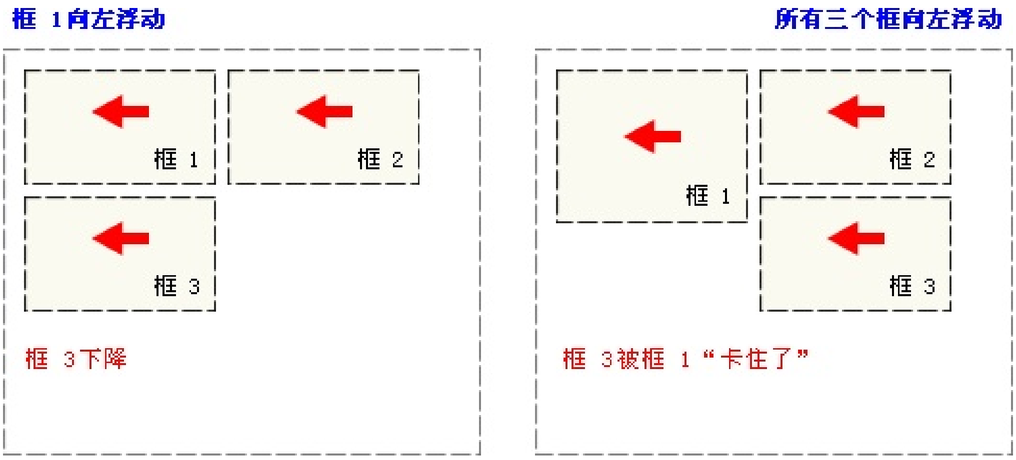


再请看下图，当框 1 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 2，使框 2 从视图中消失。

如果把所有三个框都向左移动，那么框 1 向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框。



如下图所示，如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”：



CSS float 属性

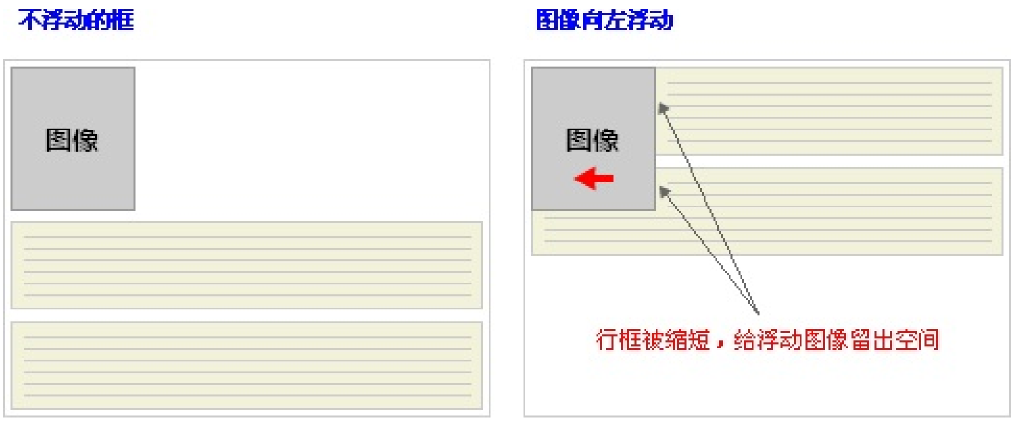
在 CSS 中，我们通过 float 属性实现元素的浮动。

如需更多有关 float 属性的知识，请访问参考手册：CSS float 属性。

行框和清理

浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。

因此，创建浮动框可以使文本围绕图像：



要想阻止行框围绕浮动框，需要对该框应用 clear 属性。clear 属性的值可以是 left、right、both 或 none，它表示框的哪些边不应该挨着浮动框。

为了实现这种效果，在被清理的元素的上外边距上添加足够的空间，使元素的顶边缘垂直下降到浮动框下面：



这是一个有用的工具，它让周围的元素为浮动元素留出空间。

让我们更详细地看看浮动和清理。假设希望让一个图片浮动到文本块的左边，并且希望这幅图片和文本包含在另一个具有背景颜色和边框的元素中。您可能编写下面的代码：

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

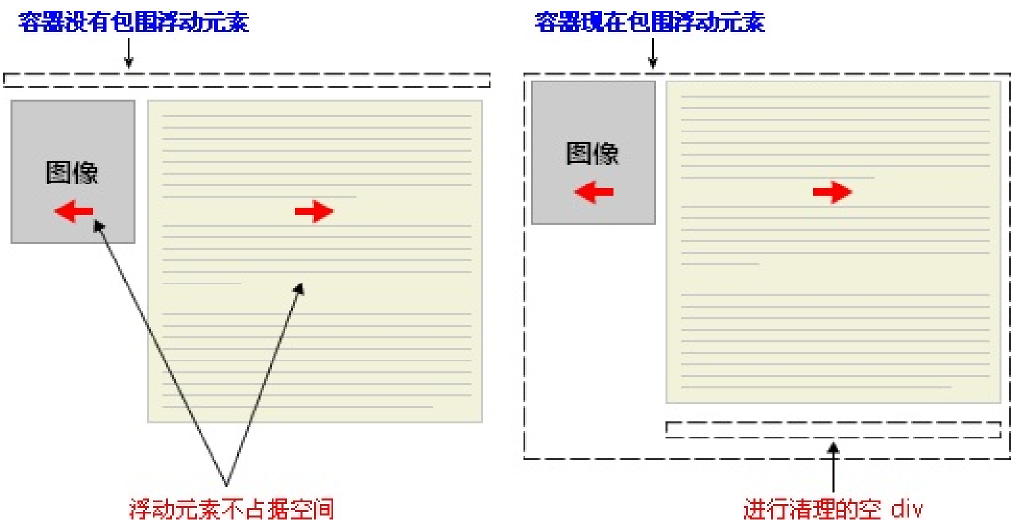
.news img {
  float: left;
  }

.news p {
  float: right;
  }

<div class="news">
<img src="news-pic.jpg" />
<p>some text</p>
</div>
```

这种情况下，出现了一个问题。因为浮动元素脱离了文档流，所以包围图片和文本的 div 不占据空间。

如何让包围元素在视觉上包围浮动元素呢？需要在这个元素中的某个地方应用 clear：



不幸的是出现了一个新的问题，由于没有现有的元素可以应用清理，所以我们只能添加一个空元素并且清理它。

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }

.clear {
  clear: both;
  }

<div class="news">
<img src="news-pic.jpg" />
<p>some text</p>
<div class="clear"></div>
</div>
```

这样可以实现我们希望的效果，但是需要添加多余的代码。常常有元素可以应用 clear，但是有时候不得不为了进行布局而添加无意义的标记。

不过我们还有另一种办法，那就是对容器 div 进行浮动：

```
.news {
  background-color: gray;
  border: solid 1px black;
  float: left;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }

<div class="news">
<img src="news-pic.jpg" />
<p>some text</p>
</div>
```

这样会得到我们希望的效果。不幸的是，下一个元素会受到这个浮动元素的影响。为了解决这个问题，有些人选择对布局中的所有东西进行浮动，然后使用适当的有意义的元素（常常是站点的页脚）对这些浮动进行清理。这有助于减少或消除不必要的标记。

事实上，W3School 站点上的所有页面都采用了这种技术，如果您打开我们使用 CSS 文件，您会看到我们对页脚的 div 进行了清理，而页脚上面的三个 div 都向左浮动。

CSS clear 属性

我们刚才详细讨论了 CSS 清理的工作原理和 clear 属性应用方法。如果您希望学习更多有关 clear 属性的知识，请访问参考手册：CSS clear 属性。

浮动和清理 实例

float 属性的简单应用

将带有边框和边界的图像浮动于段落的右侧

带标题的图像浮动于右侧

使段落的首字母浮动于左侧

创建水平菜单

创建无表格的首页

清除元素的侧面

CSS 选择器

CSS 元素选择器

CSS 元素选择器

最常见的 CSS 选择器是元素选择器。换句话说，文档的元素就是最基本的选择器。

如果设置 HTML 的样式，选择器通常将是某个 HTML 元素，比如 p、h1、em、a，甚至可以是 html 本身：

```
html {color:black;}
h1 {color:blue;}
h2 {color:silver;}
```

可以将某个样式从一个元素切换到另一个元素。

假设您决定将上面的段落文本（而不是 h1 元素）设置为灰色。只需要把 h1 选择器改为 p：

```
html {color:black;}
p {color:gray;}
h2 {color:silver;}
```

类型选择器

在 W3C 标准中，元素选择器又称为类型选择器（type selector）。

“类型选择器匹配文档语言元素类型的名称。类型选择器匹配文档树中该元素类型的每一个实例。”

下面的规则匹配文档树中所有 h1 元素：

```
h1 {font-family: sans-serif;}
```

因此，我们也可以为 XML 文档中的元素设置样式：

XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="note.css"?>
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

CSS 文档：

```
note
  {
  font-family:Verdana, Arial;
  margin-left:30px;
  }

to
  {
  font-size:28px;
  display: block;
  }

from
  {
  font-size:28px;
  display: block;
  }

heading
  {
  color: red;
  font-size:60px;
  display: block;
  }

body
  {
  color: blue;
  font-size:35px;
  display: block;
  }
```

查看效果

通过上面的例子，您可以看到，CSS 元素选择器（类型选择器）可以设置 XML 文档中元素的样式。

如果您需要更多关于为 XML 文档添加样式的知识，请访问 w3school 的 XML 教程。

CSS 分组

选择器分组

假设希望 h2 元素和段落都有灰色。为达到这个目的，最容易的做法是使用以下声明：

```
h2, p {color:gray;}
```

将 h2 和 p 选择器放在规则左边，然后用逗号分隔，就定义了一个规则。其右边的样式（color:gray;）将应用到这两个选择器所引用的元素。逗号告诉浏览器，规则中包含两个不同的选择器。如果没有这个逗号，那么规则的含义将完全不同。参见后代选择器。

可以将任意多个选择器分组在一起，对此没有任何限制。

例如，如果您想把很多元素显示为灰色，可以使用类似如下的规则：

```
body, h2, p, table, th, td, pre, strong, em {color:gray;}
```

提示：通过分组，创作者可以将某些类型的样式"压缩"在一起，这样就可以得到更简洁的样式表。

以下的两组规则能得到同样的结果，不过可以很清楚地看出哪一个写起来更容易：

```
/* no grouping */
h1 {color:blue;}
h2 {color:blue;}
h3 {color:blue;}
h4 {color:blue;}
h5 {color:blue;}
h6 {color:blue;}

/* grouping */
h1, h2, h3, h4, h5, h6 {color:blue;}
```

分组提供了一些有意思的选择。例如，下例中的所有规则分组都是等价的，每个组只是展示了对选择器和声明分组的不同方法：

```
/* group 1 */
h1 {color:silver; background:white;}
h2 {color:silver; background:gray;}
h3 {color:white; background:gray;}
h4 {color:silver; background:white;}
b {color:gray; background:white;}

/* group 2 */
h1, h2, h4 {color:silver;}
h2, h3 {background:gray;}
h1, h4, b {background:white;}
h3 {color:white;}
b {color:gray;}

/* group 3 */
h1, h4 {color:silver; background:white;}
h2 {color:silver;}
h3 {color:white;}
h2, h3 {background:gray;}
b {color:gray; background:white;}
```

亲自试一试：

- group 1
- group 2
- group 3

请注意，group 3 中使用了“声明分组”。稍后我们会为您介绍“声明分组”。

通配符选择器

CSS2 引入了一种新的简单选择器 - 通配选择器（universal selector），显示为一个星号（*）。该选择器可以与任何元素匹配，就像是一个通配符。

例如，下面的规则可以使文档中的每个元素都为红色：

```
* {color:red;}
```

这个声明等价于列出了文档中所有元素的一个分组选择器。利用通配选择器，只需敲一次键（仅一个星号）就能使文档中所有元素的 color 属性值指定为 red。

声明分组

我们既可以对选择器进行分组，也可以对声明分组。

假设您希望所有 h1 元素都有红色背景，并使用 28 像素高的 Verdana 字体显示为蓝色文本，可以写以下样式：

```
h1 {font: 28px Verdana;}
h1 {color: blue;}
h1 {background: red;}
```

但是上面这种做法的效率并不高。尤其是当我们为一个有多个样式的元素创建这样一个列表时会很麻烦。相反，我们可以将声明分组在一起：

```
h1 {font: 28px Verdana; color: white; background: black;}
```

这与前面的 3 行样式表的效果完全相同。

注意，对声明分组，一定要在各个声明的最后使用分号，这很重要。浏览器会忽略样式表中的空白符。只要加了分号，就可以毫无顾忌地采用以下格式建立样式：

```
h1 {
  font: 28px Verdana;
  color: blue;
  background: red;
  }
```

怎么样，上面这种写法的可读性是不是更强。

不过，如果忽略了第二个分号，用户代理就会把这个样式表解释如下：

```
h1 {
  font: 28px Verdana;
  color: blue background: red;
  }
```

因为 background 对 color 来说不是一个合法值，而且由于只能为 color 指定一个关键字，所以用户代理会完全忽略这个 color 声明（包括 background: black 部分）。这样 h1 标题只会显示为蓝色，而没有红色背景，不过更有可能根本得不到蓝色的 h1。相反，这些标题只会显示为默认颜色（通常是黑色），而且根本没有背景色。font: 28px Verdana 声明仍能正常发挥作用，因为它确实正确地以一个分号结尾。

与选择器分组一样，声明分组也是一种便利的方法，可以缩短样式表，使之更清晰，也更易维护。

提示：在规则的最后一个声明后也加上分号是一个好习惯。在向规则增加另一个声明时，就不必担心忘记再插入一个分号。

结合选择器和声明的分组

我们可以在一个规则中结合选择器分组和声明分组，就可以使用很少的语句定义相对复杂的样式。

下面的规则为所有标题指定了一种复杂的样式：

```
h1, h2, h3, h4, h5, h6 {
  color:gray;
  background: white;
  padding: 10px;
  border: 1px solid black;
  font-family: Verdana;
  }
```

上面这条规则将所有标题的样式定义为带有白色背景的灰色文本，其内边距是 10 像素，并带有 1 像素的实心边框，文本字体是 Verdana。

CSS 类选择器详解

类选择器允许以一种独立于文档元素的方式来指定样式。

CSS 类选择器

类选择器允许以一种独立于文档元素的方式来指定样式。

该选择器可以单独使用，也可以与其他元素结合使用。

提示：只有适当地标记文档后，才能使用这些选择器，所以使用这两种选择器通常需要先做一些构想和计划。

要应用样式而不考虑具体设计的元素，最常用的方法就是使用类选择器。

修改 HTML 代码

在使用类选择器之前，需要修改具体的文档标记，以便类选择器正常工作。

为了将类选择器的样式与元素关联，必须将 class 指定为一个适当的值。请看下面的 HTML 代码：

```
<h1 class="important">
This heading is very important.
</h1>

<p class="important">
This paragraph is very important.
</p>
```

在上面的代码中，两个元素的 class 都指定为 important：第一个标题（ h1 元素），第二个段落（p 元素）。

语法

然后我们使用以下语法向这些归类的元素应用样式，即类名前有一个点号（.），然后结合通配选择器：

```
*.important {color:red;}
```

如果您只想选择所有类名相同的元素，可以在类选择器中忽略通配选择器，这没有任何不好的影响：

```
.important {color:red;}
```

结合元素选择器

类选择器可以结合元素选择器来使用。

例如，您可能希望只有段落显示为红色文本：

```
p.important {color:red;}
```

选择器现在会匹配 class 属性包含 important 的所有 p 元素，但是其他任何类型的元素都不匹配，不论是否有此 class 属性。选择器 p.important 解释为：“其 class 属性值为 important 的所有段落”。 因为 h1 元素不是段落，这个规则的选择器与之不匹配，因此 h1 元素不会变成红色文本。

如果你确实希望为 h1 元素指定不同的样式，可以使用选择器 h1.important：

```
p.important {color:red;}
h1.important {color:blue;}
```

CSS 多类选择器

在上一节中，我们处理了 class 值中包含一个词的情况。在 HTML 中，一个 class 值中可能包含一个词列表，各个词之间用空格分隔。例如，如果希望将一个特定的元素同时标记为重要（important）和警告（warning），就可以写作：

```
<p class="important warning">
This paragraph is a very important warning.
</p>
```

这两个词的顺序无关紧要，写成 warning important 也可以。

我们假设 class 为 important 的所有元素都是粗体，而 class 为 warning 的所有元素为斜体，class 中同时包含 important 和 warning 的所有元素还有一个银色的背景 。就可以写作：

```
.important {font-weight:bold;}
.warning {font-style:italic;}
.important.warning {background:silver;}
```

通过把两个类选择器链接在一起，仅可以选择同时包含这些类名的元素（类名的顺序不限）。

如果一个多类选择器包含类名列表中没有的一个类名，匹配就会失败。请看下面的规则：

```
.important.urgent {background:silver;}
```

不出所料，这个选择器将只匹配 class 属性中包含词 important 和 urgent 的 p 元素。因此，如果一个 p 元素的 class 属性中只有词 important 和 warning，将不能匹配。不过，它能匹配以下元素：

```
<p class="important urgent warning">
This paragraph is a very important and urgent warning.
</p>
```

重要事项：在 IE7 之前的版本中，不同平台的 Internet Explorer 都不能正确地处理多类选择器。

CSS ID 选择器详解

ID 选择器允许以一种独立于文档元素的方式来指定样式。

CSS ID 选择器

在某些方面，ID 选择器类似于类选择器，不过也有一些重要差别。

语法

首先，ID 选择器前面有一个 # 号 - 也称为棋盘号或井号。

请看下面的规则：

```
*#intro {font-weight:bold;}
```

与类选择器一样，ID 选择器中可以忽略通配选择器。前面的例子也可以写作：

```
#intro {font-weight:bold;}
```

这个选择器的效果将是一样的。

第二个区别是 ID 选择器不引用 class 属性的值，毫无疑问，它要引用 id 属性中的值。

以下是一个实际 ID 选择器的例子：

```
<p id="intro">This is a paragraph of introduction.</p>
```

类选择器还是 ID 选择器？

在类选择器这一章中我们曾讲解过，可以为任意多个元素指定类。前一章中类名 important 被应用到 p 和 h1 元素，而且它还可以应用到更多元素。

区别 1：只能在文档中使用一次

与类不同，在一个 HTML 文档中，ID 选择器会使用一次，而且仅一次。

区别 2：不能使用 ID 词列表

不同于类选择器，ID 选择器不能结合使用，因为 ID 属性不允许有以空格分隔的词列表。

区别 3：ID 能包含更多含义

类似于类，可以独立于元素来选择 ID。有些情况下，您知道文档中会出现某个特定 ID 值，但是并不知道它会出现在哪个元素上，所以您想声明独立的 ID 选择器。例如，您可能知道在一个给定的文档中会有一个 ID 值为 mostImportant 的元素。您不知道这个最重要的东西是一个段落、一个短语、一个列表项还是一个小节标题。您只知道每个文档都会有这么一个最重要的内容，它可能在任何元素中，而且只能出现一个。在这种情况下，可以编写如下规则：

```
#mostImportant {color:red; background:yellow;}
```

这个规则会与以下各个元素匹配（这些元素不能在同一个文档中同时出现，因为它们都有相同的 ID 值）：

```
<h1 id="mostImportant">This is important!</h1>
<em id="mostImportant">This is important!</em>
<ul id="mostImportant">This is important!</ul>
```

亲自试一试：

- 为 id 为 mostImportant 的 h1 元素设定样式
- 为 id 为 mostImportant 的 em 元素设定样式
- 为 id 为 mostImportant 的 ul 元素设定样式

区分大小写

请注意，类选择器和 ID 选择器可能是区分大小写的。这取决于文档的语言。HTML 和 XHTML 将类和 ID 值定义为区分大小写，所以类和 ID 值的大小写必须与文档中的相应值匹配。

因此，对于以下的 CSS 和 HTML，元素不会变成粗体：

```
#intro {font-weight:bold;}

<p id="Intro">This is a paragraph of introduction.</p>
```

由于字母 i 的大小写不同，所以选择器不会匹配上面的元素。

CSS 属性选择器详解

CSS 2 引入了属性选择器。

属性选择器可以根据元素的属性及属性值来选择元素。

简单属性选择

如果希望选择有某个属性的元素，而不论属性值是什么，可以使用简单属性选择器。

例子 1

如果您希望把包含标题（title）的所有元素变为红色，可以写作：

```
*[title] {color:red;}
```

例子 2

与上面类似，可以只对有 href 属性的锚（a 元素）应用样式：

```
a[href] {color:red;}
```

例子 3

还可以根据多个属性进行选择，只需将属性选择器链接在一起即可。

例如，为了将同时有 href 和 title 属性的 HTML 超链接的文本设置为红色，可以这样写：

```
a[href][title] {color:red;}
```

例子 4

可以采用一些创造性的方法使用这个特性。

例如，可以对所有带有 alt 属性的图像应用样式，从而突出显示这些有效的图像：

```
img[alt] {border: 5px solid red;}
```

提示：上面这个特例更适合用来诊断而不是设计，即用来确定图像是否确实有效。

例子 5：为 XML 文档使用属性选择器

属性选择器在 XML 文档中相当有用，因为 XML 语言主张要针对元素和属性的用途指定元素名和属性名。

假设我们为描述太阳系行星设计了一个 XML 文档。如果我们想选择有 moons 属性的所有 planet 元素，使之显示为红色，以便能更关注有 moons 的行星，就可以这样写：

```
planet[moons] {color:red;}
```

这会让以下标记片段中的第二个和第三个元素的文本显示为红色，但第一个元素的文本不是红色：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

根据具体属性值选择

除了选择拥有某些属性的元素，还可以进一步缩小选择范围，只选择有特定属性值的元素。

例子 1

例如，假设希望将指向 Web 服务器上某个指定文档的超链接变成红色，可以这样写：

```
a[href="http://www.w3school.com.cn/about_us.asp"] {color: red;}
```

例子 2

与简单属性选择器类似，可以把多个属性-值选择器链接在一起来选择一个文档。

```
a[href="http://www.w3school.com.cn/"][title="W3School"] {color: rec
```

这会把以下标记中的第一个超链接的文本变为红色，但是第二个或第三个链接不受影响：

```
<a href="http://www.w3school.com.cn/" title="W3School">W3School</a>
<a href="http://www.w3school.com.cn/css/" title="CSS">CSS</a>
<a href="http://www.w3school.com.cn/html/" title="HTML">HTML</a>
```

例子 3

同样地，XML 语言也可以利用这种方法来设置样式。

下面我们再回到行星那个例子中。假设只希望选择 moons 属性值为 1 的那些 planet 元素：

```
planet[moons="1"] {color: red;}
```

上面的代码会把以下标记中的第二个元素变成红色，但第一个和第三个元素不受影响：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

属性与属性值必须完全匹配

请注意，这种格式要求必须与属性值完全匹配。

如果属性值包含用空格分隔的值列表，匹配就可能出问题。

请考虑一下的标记片段：

```
<p class="important warning">This paragraph is a very important wa
```

如果写成 p[class="important"]，那么这个规则不能匹配示例标记。

要根据具体属性值来选择该元素，必须这样写：

```
p[class="important warning"] {color: red;}
```

根据部分属性值选择

如果需要根据属性值中的词列表的某个词进行选择，则需要使用波浪号（~）。

假设您想选择 class 属性中包含 important 的元素，可以用下面这个选择器做到这一点：

```
p[class~="important"] {color: red;}
```

如果忽略了波浪号，则说明需要完成完全值匹配。

部分值属性选择器与点号类名记法的区别

该选择器等价于我们在类选择器中讨论过的点号类名记法。

也就是说，p.important 和 p[class="important"] 应用到 HTML 文档时是等价的。

那么，为什么还要有 "~=" 属性选择器呢？因为它能用于任何属性，而不只是 class。

例如，可以有一个包含大量图像的文档，其中只有一部分是图片。对此，可以使用一个基于 title 文档的部分属性选择器，只选择这些图片：

```
img[title~="Figure"] {border: 1px solid gray;}
```

这个规则会选择 title 文本包含 "Figure" 的所有图像。没有 title 属性或者 title 属性中不包含 "Figure" 的图像都不会匹配。

子串匹配属性选择器

下面为您介绍一个更高级的选择器模块，它是 CSS2 完成之后发布的，其中包含了更多的部分值属性选择器。按照规范的说法，应该称之为“子串匹配属性选择器”。

很多现代浏览器都支持这些选择器，包括 IE7。

下表是对这些选择器的简单总结：

类型	描述
[abc^="def"]	选择 abc 属性值以 "def" 开头的所有元素
[abc$="def"]	选择 abc 属性值以 "def" 结尾的所有元素
[abc*="def"]	选择 abc 属性值中包含子串 "def" 的所有元素

可以想到，这些选择有很多用途。

举例来说，如果希望对指向 W3School 的所有链接应用样式，不必为所有这些链接指定 class，再根据这个类编写样式，而只需编写以下规则：

```
a[href*="w3school.com.cn"] {color: red;}
```

提示：任何属性都可以使用这些选择器。

特定属性选择类型

最后为您介绍特定属性选择器。请看下面的例子：

```
*[lang|="en"] {color: red;}
```

上面这个规则会选择 lang 属性等于 en 或以 en- 开头的所有元素。因此，以下示例标记中的前三个元素将被选中，而不会选择后两个元素：

```
<p lang="en">Hello!</p>
<p lang="en-us">Greetings!</p>
<p lang="en-au">G'day!</p>
<p lang="fr">Bonjour!</p>
<p lang="cy-en">Jrooana!</p>
```

一般来说，[att|="val"] 可以用于任何属性及其值。

假设一个 HTML 文档中有一系列图片，其中每个图片的文件名都形如 *figure-1.jpg* 和 *figure-2.jpg*。就可以使用以下选择器匹配所有这些图像：

```
img[src|="figure"] {border: 1px solid gray;}
```

当然，这种属性选择器最常见的用途还是匹配语言值。

CSS 选择器参考手册

选择器	描述
[attribute]	用于选取带有指定属性的元素。
[attribute=value]	用于选取带有指定属性和值的元素。
[attribute~=value]	用于选取属性值中包含指定词汇的元素。
[attribute|=value]	用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

CSS 后代选择器

后代选择器（**descendant selector**）又称为包含选择器。

后代选择器可以选择作为某元素后代的元素。

根据上下文选择元素

我们可以定义后代选择器来创建一些规则，使这些规则在某些文档结构中起作用，而在另外一些结构中不起作用。

举例来说，如果您希望只对 h1 元素中的 em 元素应用样式，可以这样写：

```
h1 em {color:red;}
```

上面这个规则会把作为 h1 元素后代的 em 元素的文本变为 红色。其他 em 文本（如段落或块引用中的 em）则不会被这个规则选中：

```
<h1>This is a <em>important</em> heading</h1>
<p>This is a <em>important</em> paragraph.</p>
```

当然，您也可以在 h1 中找到的每个 em 元素上放一个 class 属性，但是显然，后代选择器的效率更高。

语法解释

在后代选择器中，规则左边的选择器一端包括两个或多个用空格分隔的选择器。选择器之间的空格是一种结合符（combinator）。每个空格结合符可以解释为“... 在 ... 找到”、“... 作为 ... 的一部分”、“... 作为 ... 的后代”，但是要求必须从右向左读选择器。

因此，h1 em 选择器可以解释为 “作为 h1 元素后代的任何 em 元素”。如果要从左向右读选择器，可以换成以下说法：“包含 em 的所有 h1 会把以下样式应用到该 em”。

具体应用

后代选择器的功能极其强大。有了它，可以使 HTML 中不可能实现的任务成为可能。

假设有一个文档，其中有一个边栏，还有一个主区。边栏的背景为蓝色，主区的背景为白色，这两个区都包含链接列表。不能把所有链接都设置为蓝色，因为这样一来边栏中的蓝色链接都无法看到。

解决方法是使用后代选择器。在这种情况下，可以为包含边栏的 div 指定值为 sidebar 的 class 属性，并把主区的 class 属性值设置为 maincontent。然后编写以下样式：

```
div.sidebar {background:blue;}
div.maincontent {background:white;}
div.sidebar a:link {color:white;}
div.maincontent a:link {color:blue;}
```

有关后代选择器有一个易被忽视的方面，即两个元素之间的层次间隔可以是无限的。

例如，如果写作 ul em，这个语法就会选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。

因此，ul em 将会选择以下标记中的所有 em 元素：

```
<ul>
  <li>List item 1
    <ol>
      <li>List item 1-1</li>
      <li>List item 1-2</li>
      <li>List item 1-3
        <ol>
          <li>List item 1-3-1</li>
          <li>List item <em>1-3-2</em></li>
          <li>List item 1-3-3</li>
        </ol>
      </li>
      <li>List item 1-4</li>
    </ol>
  </li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

CSS 属性选择器详解

CSS 2 引入了属性选择器。

属性选择器可以根据元素的属性及属性值来选择元素。

简单属性选择

如果希望选择有某个属性的元素，而不论属性值是什么，可以使用简单属性选择器。

例子 1

如果您希望把包含标题（title）的所有元素变为红色，可以写作：

```
*[title] {color:red;}
```

例子 2

与上面类似，可以只对有 href 属性的锚（a 元素）应用样式：

```
a[href] {color:red;}
```

例子 3

还可以根据多个属性进行选择，只需将属性选择器链接在一起即可。

例如，为了将同时有 href 和 title 属性的 HTML 超链接的文本设置为红色，可以这样写：

```
a[href][title] {color:red;}
```

例子 4

可以采用一些创造性的方法使用这个特性。

例如，可以对所有带有 alt 属性的图像应用样式，从而突出显示这些有效的图像：

```
img[alt] {border: 5px solid red;}
```

提示：上面这个特例更适合用来诊断而不是设计，即用来确定图像是否确实有效。

例子 5：为 XML 文档使用属性选择器

属性选择器在 XML 文档中相当有用，因为 XML 语言主张要针对元素和属性的用途指定元素名和属性名。

假设我们为描述太阳系行星设计了一个 XML 文档。如果我们想选择有 moons 属性的所有 planet 元素，使之显示为红色，以便能更关注有 moons 的行星，就可以这样写：

```
planet[moons] {color:red;}
```

这会让以下标记片段中的第二个和第三个元素的文本显示为红色，但第一个元素的文本不是红色：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

根据具体属性值选择

除了选择拥有某些属性的元素，还可以进一步缩小选择范围，只选择有特定属性值的元素。

例子 1

例如，假设希望将指向 Web 服务器上某个指定文档的超链接变成红色，可以这样写：

```
a[href="http://www.w3school.com.cn/about_us.asp"] {color: red;}
```

例子 2

与简单属性选择器类似，可以把多个属性-值选择器链接在一起来选择一个文档。

```
a[href="http://www.w3school.com.cn/"][title="W3School"] {color: rec
```

这会把以下标记中的第一个超链接的文本变为红色，但是第二个或第三个链接不受影响：

```
<a href="http://www.w3school.com.cn/" title="W3School">W3School</a>
<a href="http://www.w3school.com.cn/css/" title="CSS">CSS</a>
<a href="http://www.w3school.com.cn/html/" title="HTML">HTML</a>
```

例子 3

同样地，XML 语言也可以利用这种方法来设置样式。

下面我们再回到行星那个例子中。假设只希望选择 moons 属性值为 1 的那些 planet 元素：

```
planet[moons="1"] {color: red;}
```

上面的代码会把以下标记中的第二个元素变成红色，但第一个和第三个元素不受影响：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

属性与属性值必须完全匹配

请注意，这种格式要求必须与属性值完全匹配。

如果属性值包含用空格分隔的值列表，匹配就可能出问题。

请考虑一下的标记片段：

```
<p class="important warning">This paragraph is a very important wa
```

如果写成 p[class="important"]，那么这个规则不能匹配示例标记。

要根据具体属性值来选择该元素，必须这样写：

```
p[class="important warning"] {color: red;}
```

根据部分属性值选择

如果需要根据属性值中的词列表的某个词进行选择，则需要使用波浪号（~）。

假设您想选择 class 属性中包含 important 的元素，可以用下面这个选择器做到这一点：

```
p[class~="important"] {color: red;}
```

如果忽略了波浪号，则说明需要完成完全值匹配。

部分值属性选择器与点号类名记法的区别

该选择器等价于我们在类选择器中讨论过的点号类名记法。

也就是说，p.important 和 p[class="important"] 应用到 HTML 文档时是等价的。

那么，为什么还要有 "~=" 属性选择器呢？因为它能用于任何属性，而不只是 class。

例如，可以有一个包含大量图像的文档，其中只有一部分是图片。对此，可以使用一个基于 title 文档的部分属性选择器，只选择这些图片：

```
img[title~="Figure"] {border: 1px solid gray;}
```

这个规则会选择 title 文本包含 "Figure" 的所有图像。没有 title 属性或者 title 属性中不包含 "Figure" 的图像都不会匹配。

子串匹配属性选择器

下面为您介绍一个更高级的选择器模块，它是 CSS2 完成之后发布的，其中包含了更多的部分值属性选择器。按照规范的说法，应该称之为“子串匹配属性选择器”。

很多现代浏览器都支持这些选择器，包括 IE7。

下表是对这些选择器的简单总结：

类型	描述
[abc^="def"]	选择 abc 属性值以 "def" 开头的所有元素
[abc$="def"]	选择 abc 属性值以 "def" 结尾的所有元素
[abc*="def"]	选择 abc 属性值中包含子串 "def" 的所有元素

可以想到，这些选择有很多用途。

举例来说，如果希望对指向 W3School 的所有链接应用样式，不必为所有这些链接指定 class，再根据这个类编写样式，而只需编写以下规则：

```
a[href*="w3school.com.cn"] {color: red;}
```

提示：任何属性都可以使用这些选择器。

特定属性选择类型

最后为您介绍特定属性选择器。请看下面的例子：

```
*[lang|="en"] {color: red;}
```

上面这个规则会选择 lang 属性等于 en 或以 en- 开头的所有元素。因此，以下示例标记中的前三个元素将被选中，而不会选择后两个元素：

```
<p lang="en">Hello!</p>
<p lang="en-us">Greetings!</p>
<p lang="en-au">G'day!</p>
<p lang="fr">Bonjour!</p>
<p lang="cy-en">Jrooana!</p>
```

一般来说，[att|="val"] 可以用于任何属性及其值。

假设一个 HTML 文档中有一系列图片，其中每个图片的文件名都形如 *figure-1.jpg* 和 *figure-2.jpg*。就可以使用以下选择器匹配所有这些图像：

```
img[src|="figure"] {border: 1px solid gray;}
```

当然，这种属性选择器最常见的用途还是匹配语言值。

CSS 选择器参考手册

选择器	描述
[attribute]	用于选取带有指定属性的元素。
[attribute=value]	用于选取带有指定属性和值的元素。
[attribute~=value]	用于选取属性值中包含指定词汇的元素。
[attribute|=value]	用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

CSS 后代选择器

后代选择器（**descendant selector**）又称为包含选择器。

后代选择器可以选择作为某元素后代的元素。

根据上下文选择元素

我们可以定义后代选择器来创建一些规则，使这些规则在某些文档结构中起作用，而在另外一些结构中不起作用。

举例来说，如果您希望只对 h1 元素中的 em 元素应用样式，可以这样写：

```
h1 em {color:red;}
```

上面这个规则会把作为 h1 元素后代的 em 元素的文本变为 红色。其他 em 文本（如段落或块引用中的 em）则不会被这个规则选中：

```
<h1>This is a <em>important</em> heading</h1>
<p>This is a <em>important</em> paragraph.</p>
```

当然，您也可以在 h1 中找到的每个 em 元素上放一个 class 属性，但是显然，后代选择器的效率更高。

语法解释

在后代选择器中，规则左边的选择器一端包括两个或多个用空格分隔的选择器。选择器之间的空格是一种结合符（combinator）。每个空格结合符可以解释为“... 在 ... 找到”、“... 作为 ... 的一部分”、“... 作为 ... 的后代”，但是要求必须从右向左读选择器。

因此，h1 em 选择器可以解释为 “作为 h1 元素后代的任何 em 元素”。如果要从左向右读选择器，可以换成以下说法：“包含 em 的所有 h1 会把以下样式应用到该 em”。

具体应用

后代选择器的功能极其强大。有了它，可以使 HTML 中不可能实现的任务成为可能。

假设有一个文档，其中有一个边栏，还有一个主区。边栏的背景为蓝色，主区的背景为白色，这两个区都包含链接列表。不能把所有链接都设置为蓝色，因为这样一来边栏中的蓝色链接都无法看到。

解决方法是使用后代选择器。在这种情况下，可以为包含边栏的 div 指定值为 sidebar 的 class 属性，并把主区的 class 属性值设置为 maincontent。然后编写以下样式：

```
div.sidebar {background:blue;}
div.maincontent {background:white;}
div.sidebar a:link {color:white;}
div.maincontent a:link {color:blue;}
```

有关后代选择器有一个易被忽视的方面，即两个元素之间的层次间隔可以是无限的。

例如，如果写作 ul em，这个语法就会选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。

因此，ul em 将会选择以下标记中的所有 em 元素：

```
<ul>
  <li>List item 1
    <ol>
      <li>List item 1-1</li>
      <li>List item 1-2</li>
      <li>List item 1-3
        <ol>
          <li>List item 1-3-1</li>
          <li>List item <em>1-3-2</em></li>
          <li>List item 1-3-3</li>
        </ol>
      </li>
      <li>List item 1-4</li>
    </ol>
  </li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

CSS 子元素选择器

与后代选择器相比，子元素选择器（**Child selectors**）只能选择作为某元素子元素的元素。

选择子元素

如果您不希望选择任意的后代元素，而是希望缩小范围，只选择某个元素的子元素，请使用子元素选择器（Child selector）。

例如，如果您希望选择只作为 h1 元素子元素的 strong 元素，可以这样写：

```
h1 > strong {color:red;}
```

这个规则会把第一个 h1 下面的两个 strong 元素变为红色，但是第二个 h1 中的 strong 不受影响：

```
<h1>This is <strong>very</strong> <strong>very</strong> important.<
<h1>This is <em>really <strong>very</strong></em> important.</h1>
```

语法解释

您应该已经注意到了，子选择器使用了大于号（子结合符）。

子结合符两边可以有空白符，这是可选的。因此，以下写法都没有问题：

```
h1 > strong
h1> strong
h1 >strong
h1>strong
```

如果从右向左读，选择器 h1 > strong 可以解释为“选择作为 h1 元素子元素的所有 strong 元素”。

结合后代选择器和子选择器

请看下面这个选择器：

```
table.company td > p
```

上面的选择器会选择作为 td 元素子元素的所有 p 元素，这个 td 元素本身从 table 元素继承，该 table 元素有一个包含 company 的 class 属性。

CSS 相邻兄弟选择器

相邻兄弟选择器（**Adjacent sibling selector**）可选择紧接在另一元素后的元素，且二者有相同父元素。

选择相邻兄弟

如果需要选择紧接在另一个元素后的元素，而且二者有相同的父元素，可以使用相邻兄弟选择器（Adjacent sibling selector）。

例如，如果要增加紧接在 h1 元素后出现的段落的上边距，可以这样写：

```
h1 + p {margin-top:50px;}
```

这个选择器读作："选择紧接在 h1 元素后出现的段落，h1 和 p 元素拥有共同的父元素"。

语法解释

相邻兄弟选择器使用了加号（+），即相邻兄弟结合符（Adjacent sibling combinator）。

注释：与子结合符一样，相邻兄弟结合符旁边可以有空白符。

请看下面这个文档树片段：

```
<div>
  <ul>
    <li>List item 1</li>
    <li>List item 2</li>
    <li>List item 3</li>
  </ul>
  <ol>
    <li>List item 1</li>
    <li>List item 2</li>
    <li>List item 3</li>
  </ol>
</div>
```

在上面的片段中，div 元素中包含两个列表：一个无序列表，一个有序列表，每个列表都包含三个列表项。这两个列表是相邻兄弟，列表项本身也是相邻兄弟。不过，第一个列表中的列表项与第二个列表中的列表项不是相邻兄弟，因为这两组列表项不属于同一父元素（最多只能算堂兄弟）。

请记住，用一个结合符只能选择两个相邻兄弟中的第二个元素。请看下面的选择器：

```
li + li {font-weight:bold;}
```

上面这个选择器只会把列表中的第二个和第三个列表项变为粗体。第一个列表项不受影响。

结合其他选择器

相邻兄弟结合符还可以结合其他结合符：

```
html > body table + ul {margin-top:20px;}
```

这个选择器解释为：选择紧接在 table 元素后出现的所有兄弟 ul 元素，该 table 元素包含在一个 body 元素中，body 元素本身是 html 元素的子元素。

CSS 伪类 (Pseudo-classes)

CSS 伪类用于向某些选择器添加特殊的效果。

CSS 伪类 (Pseudo-classes)实例：

超链接

```
&lt;html&gt;
&lt;head&gt;

&lt;style type="text/css"&gt;
a:link {color: #FF0000}
a:visited {color: #00FF00}
a:hover {color: #FF00FF}
a:active {color: #0000FF}
&lt;/style&gt;

&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;&lt;b&gt;&lt;a href="/index.html" target="_blank"&gt;这是一
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;在 CSS 定义中，a:hover 必须位于 a:lin
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;在 CSS 定义中，a:active 必须位于 a:ho

&lt;/body&gt;
&lt;/html&gt;
```

超链接 2

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
a.one:link {color: #ff0000}
a.one:visited {color: #0000ff}
a.one:hover {color: #ffcc00}

a.two:link {color: #ff0000}
a.two:visited {color: #0000ff}
a.two:hover {font-size: 150%}

a.three:link {color: #ff0000}
a.three:visited {color: #0000ff}
a.three:hover {background: #66ff66}

a.four:link {color: #ff0000}
a.four:visited {color: #0000ff}
a.four:hover {font-family: monospace}

a.five:link {color: #ff0000; text-decoration: none}
a.five:visited {color: #0000ff; text-decoration: none}
a.five:hover {text-decoration: underline}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;请把鼠标移动到这些链接上，以查看效果：&lt;/p&gt;

&lt;p&gt;&lt;b&gt;&lt;a class="one" href="/index.html" target="_bla
&lt;p&gt;&lt;b&gt;&lt;a class="two" href="/index.html" target="_bla
&lt;p&gt;&lt;b&gt;&lt;a class="three" href="/index.html" target="_b
&lt;p&gt;&lt;b&gt;&lt;a class="four" href="/index.html" target="_bl
&lt;p&gt;&lt;b&gt;&lt;a class="five" href="/index.html" target="_bl
&lt;/body&gt;

&lt;/html&gt;
```

超链接 - :focus 的使用

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
input:focus
{
background-color:yellow;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;form action="form_action.asp" method="get"&gt;
First name: &lt;input type="text" name="fname" /&gt;&lt;br /&gt;
Last name: &lt;input type="text" name="lname" /&gt;&lt;br /&gt;
&lt;input type="submit" value="Submit" /&gt;
&lt;/form&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;如果已规定 !DOCTYPE，那么 Internet E

&lt;/body&gt;
&lt;/html&gt;
```

:first-child（首个子对象）

```
&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p:first-child {font-weight: bold;}
li:first-child {text-transform:uppercase;}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;div&gt;
&lt;p&gt;These are the necessary steps:&lt;/p&gt;
&lt;ul&gt;
&lt;li&gt;Intert Key&lt;/li&gt;
&lt;li&gt;Turn key &lt;strong&gt;clockwise&lt;/strong&gt;&lt;/li&g
&lt;li&gt;Push accelerator&lt;/li&gt;
&lt;/ul&gt;
&lt;p&gt;Do &lt;em&gt;not&lt;/em&gt; push the brake at the same ti
&lt;/div&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;必须声明 DOCTYPE，这样 :first-child
&lt;/body&gt;

&lt;/html&gt;
```

:lang（语言）

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
q:lang(no)
{
quotes: "~" "~"
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;:lang 伪类允许您为不同的语言定义特殊的规则。在下面的例子中，在下面的

&lt;p&gt;一些文本 &lt;q lang="no"&gt;段落中的引用&lt;/q&gt; 一些文本。&
&lt;/body&gt;

&lt;/html&gt;
```

语法

伪类的语法：

```
selector : pseudo-class {property: value}
```

CSS 类也可与伪类搭配使用。

```
selector.class : pseudo-class {property: value}
```

锚伪类

在支持 CSS 的浏览器中，链接的不同状态都可以不同的方式显示，这些状态包括：活动状态，已被访问状态，未被访问状态，和鼠标悬停状态。

```
a:link {color: #FF0000}        /* 未访问的链接 */
a:visited {color: #00FF00}    /* 已访问的链接 */
a:hover {color: #FF00FF}    /* 鼠标移动到链接上 */
a:active {color: #0000FF}    /* 选定的链接 */
```

提示：在 CSS 定义中，a:hover 必须被置于 a:link 和 a:visited 之后，才是有效的。

提示：在 CSS 定义中，a:active 必须被置于 a:hover 之后，才是有效的。

提示：伪类名称对大小写不敏感。

伪类与 CSS 类

伪类可以与 CSS 类配合使用：

```
a.red : visited {color: #FF0000}

<a class="red" href="css_syntax.asp">CSS Syntax</a>
```

假如上面的例子中的链接被访问过，那么它将显示为红色。

CSS2 - :first-child 伪类

您可以使用 :first-child 伪类来选择元素的第一个子元素。这个特定伪类很容易遭到误解，所以有必要举例来说明。考虑以下标记：

```
<div>
<p>These are the necessary steps:</p>
<ul>
<li>Intert Key</li>
<li>Turn key <strong>clockwise</strong></li>
<li>Push accelerator</li>
</ul>
<p>Do <em>not</em> push the brake at the same time as the accelerat
</div>
```

在上面的例子中，作为第一个元素的元素包括第一个 p、第一个 li 和 strong 和 em 元素。

给定以下规则：

```
p:first-child {font-weight: bold;}
li:first-child {text-transform:uppercase;}
```

第一个规则将作为某元素第一个子元素的所有 p 元素设置为粗体。第二个规则将作为某个元素（在 HTML 中，这肯定是 ol 或 ul 元素）第一个子元素的所有 li 元素变成大写。

请访问该链接，来查看这个 :first-child 实例的效果。

提示：最常见的错误是认为 p:first-child 之类的选择器会选择 p 元素的第一个子元素。

注释：必须声明 <!DOCTYPE>，这样 :first-child 才能在 IE 中生效。

为了使您更透彻地理解 :first-child 伪类，我们另外提供了 3 个例子：

例子 1 - 匹配第一个 <p> 元素

在下面的例子中，选择器匹配作为任何元素的第一个子元素的 p 元素：

```
<html>
<head>
<style type="text/css">
p:first-child {
  color: red;
  }
</style>
</head>

<body>
<p>some text</p>
<p>some text</p>
</body>
</html>
```

例子 2 - 匹配所有 <p> 元素中的第一个 <i> 元素

在下面的例子中，选择器匹配所有 <p> 元素中的第一个 <i> 元素：

```
<html>
<head>
<style type="text/css">
p > i:first-child {
  font-weight:bold;
  }
</style>
</head>

<body>
<p>some <i>text</i>. some <i>text</i>.</p>
<p>some <i>text</i>. some <i>text</i>.</p>
</body>
</html>
```

例子 3 - 匹配所有作为第一个子元素的 <p> 元素中的所有 <i> 元素

在下面的例子中，选择器匹配所有作为元素的第一个子元素的 <p> 元素中的所有 <i> 元素：

```
<html>
<head>
<style type="text/css">
p:first-child i {
  color:blue;
  }
</style>
</head>

<body>
<p>some <i>text</i>. some <i>text</i>.</p>
<p>some <i>text</i>. some <i>text</i>.</p>
</body>
</html>
```

CSS2 - :lang 伪类

:lang 伪类使你有能力为不同的语言定义特殊的规则。在下面的例子中，:lang 类为属性值为 no 的 q 元素定义引号的类型：

```
<html>
<head>

<style type="text/css">
q:lang(no)
   {
   quotes: "~" "~"
   }
</style>

</head>

<body>
<p>文字<q lang="no">段落中的引用的文字</q>文字</p>
</body></html>
```

伪类

W3C："W3C" 列指示出该属性在哪个 CSS 版本中定义（CSS1 还是 CSS2）。

属性	描述	CSS
:active	向被激活的元素添加样式。	1
:focus	向拥有键盘输入焦点的元素添加样式。	2
:hover	当鼠标悬浮在元素上方时，向元素添加样式。	1
:link	向未被访问的链接添加样式。	1
:visited	向已被访问的链接添加样式。	1
:first-child	向元素的第一个子元素添加样式。	2
:lang	向带有指定 lang 属性的元素添加样式。	2

CSS 伪元素 (Pseudo-elements)

CSS 伪元素用于向某些选择器设置特殊效果。

语法

伪元素的语法：

```
selector:pseudo-element {property:value;}
```

CSS 类也可以与伪元素配合使用：

```
selector.class:pseudo-element {property:value;}
```

:first-line 伪元素

"first-line" 伪元素用于向文本的首行设置特殊样式。

在下面的例子中，浏览器会根据 "first-line" 伪元素中的样式对 p 元素的第一行文本进行格式化：

实例

```
p:first-line
  {
  color:#ff0000;
  font-variant:small-caps;
  }
```

注释："first-line" 伪元素只能用于块级元素。

注释：下面的属性可应用于 "first-line" 伪元素：

- font
- color
- background
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform

- line-height
- clear

:first-letter 伪元素

"first-letter" 伪元素用于向文本的首字母设置特殊样式：

```
p:first-letter
  {
  color:#ff0000;
  font-size:xx-large;
  }
```

注释："first-letter" 伪元素只能用于块级元素。

注释：下面的属性可应用于 "first-letter" 伪元素：

- font
- color
- background
- margin
- padding
- border
- text-decoration
- vertical-align (仅当 float 为 none 时)
- text-transform
- line-height
- float
- clear

伪元素和 CSS 类

伪元素可以与 CSS 类配合使用：

```
p.article:first-letter
  {
  color: #FF0000;
  }

<p class="article">This is a paragraph in an article。</p>
```

上面的例子会使所有 class 为 article 的段落的首字母变为红色。

多重伪元素

可以结合多个伪元素来使用。

在下面的例子中，段落的第一个字母将显示为红色，其字体大小为 xx-large。第一行中的其余文本将为蓝色，并以小型大写字母显示。段落中的其余文本将以默认字体大小和颜色来显示：

```
p:first-letter
  {
  color:#ff0000;
  font-size:xx-large;
  }

p:first-line
  {
  color:#0000ff;
  font-variant:small-caps;
  }
```

CSS2 - :before 伪元素

":before" 伪元素可以在元素的内容前面插入新内容。

下面的例子在每个 <h1> 元素前面插入一幅图片：

```
h1:before
  {
  content:url(logo.gif);
  }
```

CSS2 - :after 伪元素

":after" 伪元素可以在元素的内容之后插入新内容。

下面的例子在每个 <h1> 元素后面插入一幅图片：

```
h1:after
  {
  content:url(logo.gif);
  }
```

伪元素

W3C："W3C" 列指示出该属性在哪个 CSS 版本中定义（CSS1 还是 CSS2）。

属性	描述	CSS
:first-letter	向文本的第一个字母添加特殊样式。	1
:first-line	向文本的首行添加特殊样式。	1
:before	在元素之前添加内容。	2
:after	在元素之后添加内容。	2

CSS 高级

CSS 水平对齐

在 CSS 中，可以使用多种属性来水平对齐元素。

对齐块元素

块元素指的是占据全部可用宽度的元素，并且在其前后都会换行。

块元素的例子：

```
<h1>
<p>
<div>
```

对于文本对齐，请参见 CSS 文本一章。

在本教程中，我们将向您展示出于布局目的如何水平对齐块级元素。

使用 margin 属性来水平对齐

可通过将左和右外边距设置为 "auto"，来对齐块元素。

注释：除非已经声明了 !DOCTYPE，否则使用 margin:auto 在 IE8 以及更早的版本中是无效的。

把左和右外边距设置为 auto，规定的是均等地分配可用的外边距。结果就是居中的元素：

实例

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

提示：如果宽度是 100%，则对齐没有效果。

注释：在 IE5 中，对于块元素存在一个外边距处理方面的 BUG。如需使上面的例子在 IE5 中有效，请添加一些额外的代码。。

使用 position 属性进行左和右对齐

对齐元素的方法之一是使用绝对定位：

实例

```
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

注释：绝对定位元素会被从正常流中删除，并且能够交叠元素。

跨浏览器兼容性问题

当像这样对齐元素时，对 <body> 元素的外边距和内边距进行预定义是一个好主意。这样可以避免在不同的浏览器中出现可见的差异。

当使用 position 属性时，IE8 以及更早的版本存在一个问题。如果容器元素（在我们的案例中是 <div class="container">）设置了指定的宽度，并且省略了 !DOCTYPE 声明，那么 IE8 以及更早的版本会在右侧增加 17px 的外边距。这似乎是为滚动条预留的空间。当使用 position 属性时，请始终设置 !DOCTYPE 声明：

实例

```
body
{
margin:0;
padding:0;
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

使用 float 属性来进行左和右对齐

对齐元素的另一种方法是使用 float 属性：

实例

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

跨浏览器兼容性问题

当像这样对齐元素时，对 <body> 元素的外边距和内边距进行预定义是一个好主意。这样可以避免在不同的浏览器中出现可见的差异。

当使用 float 属性时，IE8 以及更早的版本存在一个问题。如果省略 !DOCTYPE 声明，那么 IE8 以及更早的版本会在右侧增加 17px 的外边距。这似乎是为滚动条预留的空间。当使用 float 属性时，请始终设置 !DOCTYPE 声明：

实例

```
body
{
margin:0;
padding:0;
}

.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

CSS 尺寸 (Dimension)

CSS 尺寸 (Dimension) 属性允许你控制元素的高度和宽度。同样，它允许你增加行间距。

CSS 尺寸实例：

使用像素值设置图像的高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img.normal
{
height: auto
}

img.big
{
height: 160px
}

img.small
{
height: 30px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img class="normal" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="big" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="small" src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比设置图像的高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img.normal
{
height: auto
}

img.big
{
height: 50%
}

img.small
{
height: 10%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img class="normal" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="big" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="small" src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
width: 300px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比来设置元素的宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
width: 50%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

设置元素的最大高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
max-height: 10px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的最大宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
max-width: 300px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比来设置元素的最大宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
max-width: 50%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的最小高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
min-height: 100px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的最小宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
min-width: 1000px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比来设置元素的最小宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
min-width: 200%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比设置行间距

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
p.small {line-height: 90%}
p.big {line-height: 200%}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
这是拥有标准行高的段落。
在大多数浏览器中默认行高大约是 110% 到 120%。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
&lt;/p&gt;

&lt;p class="small"&gt;
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
&lt;/p&gt;

&lt;p class="big"&gt;
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值设置行间距

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
p.small
  {
  line-height: 10px
  }
p.big
  {
  line-height: 30px
  }
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
这是拥有标准行高的段落。
在大多数浏览器中默认行高大约是 20px。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
&lt;/p&gt;

&lt;p class="small"&gt;
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
&lt;/p&gt;

&lt;p class="big"&gt;
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用数值来设置行间距

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
p.small
{
line-height: 0.5
}
p.big
{
line-height: 2
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
这是拥有标准行高的段落。
默认行高大约是 1。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
&lt;/p&gt;

&lt;p class="small"&gt;
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
&lt;/p&gt;

&lt;p class="big"&gt;
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

CSS 尺寸属性

CSS 尺寸属性允许你控制元素的高度和宽度。同样，还允许你增加行间距。

属性	描述
height	设置元素的高度。
line-height	设置行高。
max-height	设置元素的最大高度。
max-width	设置元素的最大宽度。
min-height	设置元素的最小高度。
min-width	设置元素的最小宽度。
width	设置元素的宽度。

CSS 分类 (Classification)

CSS 分类属性允许你规定如何以及在何处显示元素。

CSS分类(Classification)实例：

如何把元素显示为内联元素

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p {display: inline}
div {display: none}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;本例中的样式表把段落元素设置为内联元素。&lt;/p&gt;

&lt;p&gt;而 div 元素不会显示出来！&lt;/p&gt;

&lt;div&gt;div 元素的内容不会显示出来！&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;
```

如何把元素显示为块级元素

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
span
{
display: block
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;span&gt;本例中的样式表把 span 元素设置为块级元素。&lt;/span&gt;
&lt;span&gt;两个 span 元素之间产生了一个换行行为。&lt;/span&gt;

&lt;/body&gt;
&lt;/html&gt;
```

float 属性的简单应用

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
float:right
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;在下面的段落中，我们添加了一个样式为 &lt;b&gt;float:right&lt;/b&
&lt;p&gt;
&lt;img src="/i/eg_cute.gif" /&gt;
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

将带有边框和边界的图像浮动于段落的右侧

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
float:right;
border:1px dotted black;
margin:0px 0px 15px 20px;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;在下面的段落中，图像会浮动到右侧，并且添加了点状的边框。我们还为图像
&lt;p&gt;
&lt;img src="/i/eg_cute.gif" /&gt;
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

带标题的图像浮动于右侧

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
div
{
float:right;
width:120px;
margin:0 0 15px 20px;
padding:15px;
border:1px solid black;
text-align:center;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;div&gt;
&lt;img src="/i/eg_cute.gif" /&gt;&lt;br /&gt;
CSS is fun!
&lt;/div&gt;
&lt;p&gt;
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;

&lt;p&gt;
在上面的段落中，div 元素的宽度是 120 像素，它其中包含图像。div 元素浮动到右侧
&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

使段落的首字母浮动于左侧

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
span
{
float:left;
width:0.7em;
font-size:400%;
font-family:algerian,courier;
line-height:80%;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;
&lt;span&gt;T&lt;/span&gt;his is some text.
This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;

&lt;p&gt;
在上面的段落中，文本的第一个字母包含在一个 span 元素中。这个 span 元素的宽度是
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建水平菜单

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
ul
{
float:left;
width:100%;
padding:0;
margin:0;
list-style-type:none;
}
a
{
float:left;
width:7em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300}
li {display:inline}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;ul&gt;
&lt;li&gt;&lt;a href="#"&gt;Link one&lt;/a&gt;&lt;/li&gt;
&lt;li&gt;&lt;a href="#"&gt;Link two&lt;/a&gt;&lt;/li&gt;
&lt;li&gt;&lt;a href="#"&gt;Link three&lt;/a&gt;&lt;/li&gt;
&lt;li&gt;&lt;a href="#"&gt;Link four&lt;/a&gt;&lt;/li&gt;
&lt;/ul&gt;

&lt;p&gt;
在上面的例子中，我们把 ul 元素和 a 元素浮向左浮动。li 元素显示为行内元素（元素
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建无表格的首页

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
div.container
{
width:100%;
margin:0px;
```

```
border:1px solid gray;
line-height:150%;
}
div.header,div.footer
{
padding:0.5em;
color:white;
background-color:gray;
clear:left;
}
h1.header
{
padding:0;
margin:0;
}
div.left
{
float:left;
width:160px;
margin:0;
padding:1em;
}
div.content
{
margin-left:190px;
border-left:1px solid gray;
padding:1em;
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;div class="container"&gt;

&lt;div class="header"&gt;&lt;h1 class="header"&gt;W3School.com.cn&

&lt;div class="left"&gt;&lt;p&gt;"Never increase, beyond what is ne

&lt;div class="content"&gt;
&lt;h2&gt;Free Web Building Tutorials&lt;/h2&gt;
&lt;p&gt;At W3School.com.cn you will find all the Web-building tuto
from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.
&lt;p&gt;W3School.com.cn - The Largest Web Developers Site On The N

&lt;div class="footer"&gt;Copyright 2008 by YingKe Investment.&lt;/
&lt;/div&gt;

&lt;/body&gt;
&lt;/html&gt;
```

定位：相对定位

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
h2.pos_left
{
position:relative;
left:-20px
}
h2.pos_right
{
position:relative;
left:20px
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h2&gt;这是位于正常位置的标题&lt;/h2&gt;
&lt;h2 class="pos_left"&gt;这个标题相对于其正常位置向左移动&lt;/h2&gt;
&lt;h2 class="pos_right"&gt;这个标题相对于其正常位置向右移动&lt;/h2&gt;
&lt;p&gt;相对定位会按照元素的原始位置对该元素进行移动。&lt;/p&gt;
&lt;p&gt;样式 "left:-20px" 从元素的原始左侧位置减去 20 像素。&lt;/p&gt;
&lt;p&gt;样式 "left:20px" 向元素的原始左侧位置增加 20 像素。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

定位：绝对定位

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
h2.pos_abs
{
position:absolute;
left:100px;
top:150px
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h2 class="pos_abs"&gt;这是带有绝对定位的标题&lt;/h2&gt;
&lt;p&gt;通过绝对定位，元素可以放置到页面上的任何位置。下面的标题距离页面左侧
&lt;/body&gt;

&lt;/html&gt;
```

定位：固定定位

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p.one
{
position:fixed;
left:5px;
top:5px;
}
p.two
{
position:fixed;
top:30px;
right:5px;
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p class="one"&gt;一些文本。&lt;/p&gt;
&lt;p class="two"&gt;更多的文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

如何使元素不可见

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
h1.visible {visibility:visible}
h1.invisible {visibility:hidden}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h1 class="visible"&gt;这是可见的标题&lt;/h1&gt;
&lt;h1 class="invisible"&gt;这是不可见的标题&lt;/h1&gt;
&lt;/body&gt;

&lt;/html&gt;
```

把表格元素设置为 collapse（请在非 IE 的浏览器中查看）

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
tr.coll
 {
 visibility:collapse
 }
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;table border="1"&gt;
&lt;tr&gt;
&lt;td&gt;Adams&lt;/td&gt;
&lt;td&gt;John&lt;/td&gt;
&lt;/tr&gt;
&lt;tr class="coll"&gt;
&lt;td&gt;Bush&lt;/td&gt;
&lt;td&gt;George&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

改变光标

```
&lt;html&gt;

&lt;body&gt;
&lt;p&gt;请把鼠标移动到单词上，可以看到鼠标指针发生变化：&lt;/p&gt;
&lt;span style="cursor:auto"&gt;
Auto&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:crosshair"&gt;
Crosshair&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:default"&gt;
Default&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:pointer"&gt;
Pointer&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:move"&gt;
Move&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:e-resize"&gt;
e-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:ne-resize"&gt;
ne-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:nw-resize"&gt;
nw-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:n-resize"&gt;
n-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:se-resize"&gt;
se-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:sw-resize"&gt;
sw-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:s-resize"&gt;
s-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:w-resize"&gt;
w-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:text"&gt;
text&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:wait"&gt;
wait&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:help"&gt;
help&lt;/span&gt;
&lt;/body&gt;

&lt;/html&gt;
```

清除元素的侧面

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
img
  {
  float:left;
  clear:both;
  }
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;img src="/i/eg_smile.gif" /&gt;
&lt;img src="/i/eg_smile.gif" /&gt;
&lt;/body&gt;

&lt;/html&gt;
```

CSS 分类属性 (Classification)

CSS 分类属性允许你控制如何显示元素，设置图像显示于另一元素中的何处，相对于其正常位置来定位元素，使用绝对值来定位元素，以及元素的可见度。

属性	描述
clear	设置一个元素的侧面是否允许其他的浮动元素。
cursor	规定当指向某元素之上时显示的指针类型。
display	设置是否及如何显示元素。
float	定义元素在哪个方向浮动。
position	把元素放置到一个静态的、相对的、绝对的、或固定的位置中。
visibility	设置元素是否可见或不可见。

CSS 导航条

演示：导航栏

- Home
- News
- Articles
- Forum
- Contact
- About

导航栏

拥有易用的导航条对于任何网站都很重要。

通过 CSS，您能够把乏味的 HTML 菜单转换为漂亮的导航栏。

导航栏 = 链接列表

导航栏需要标准的 HTML 作为基础。

在我们的例子中，将用标准的 HTML 列表来构建导航栏。

导航栏基本上是一个链接列表，因此使用 <ul> 和 <li> 元素是非常合适的：

实例

```
<ul>
<li><a href="default.asp">Home</a></li>
<li><a href="news.asp">News</a></li>
<li><a href="contact.asp">Contact</a></li>
<li><a href="about.asp">About</a></li>
</ul>
```

现在，让我们从列表中去掉圆点和外边距：

实例

```
ul
{
list-style-type:none;
margin:0;
padding:0;
}
```

例子解释：

- list-style-type:none - 删除圆点。导航栏不需要列表项标记。
- 把外边距和内边距设置为 0 可以去除浏览器的默认设定。

以上例子中的代码是用在垂直、水平导航栏中的标准代码。

垂直导航栏

如需构建垂直导航栏，我们只需要定义 <a> 元素的样式，除了上面的代码之外：

实例

```
a
{
display:block;
width:60px;
}
```

例子解释：

- display:block - 把链接显示为块元素可使整个链接区域可点击（不仅仅是文本），同时也允许我们规定宽度。
- width:60px - 块元素默认占用全部可用宽度。我们需要规定 60 像素的宽度。

提示：请同时看一看我们完整样式的导航栏实例。

注释：请始终规定垂直导航栏中 <a> 元素的宽度。如果省略宽度，IE6 会产生意想不到的结果。

水平导航栏

有两种创建水平导航栏的方法。使用行内或浮动列表项。

两种方法都不错，但是如果您希望链接拥有相同的尺寸，就必须使用浮动方法。

行内列表项

除了上面的"标准"代码，构建水平导航栏的方法之一是将 <li> 元素规定为行内元素：

实例

```
li
{
display:inline;
}
```

例子解释：

display:inline; - 默认地， 元素是块元素。在这里，我们去除了每个列表项前后的换行，以便让它们在一行中显示。

提示：请看一下我们完整样式的水平导航栏实例。

对列表项进行浮动

在上面的例子中，链接的宽度是不同的。

为了让所有链接拥有相等的宽度，浮动 <li> 元素并规定 <a> 元素的宽度：

实例

```
li
{
float:left;
}
a
{
display:block;
width:60px;
}
```

例子解释：

- float:left - 使用 float 来把块元素滑向彼此。
- display:block - 把链接显示为块元素可使整个链接区域可点击（不仅仅是文本），同时也允许我们规定宽度。
- width:60px - 由于块元素默认占用全部可用宽度，链接无法滑动至彼此相邻。我们需要规定 60 像素的宽度。

提示：请看一下我们完整样式的水平导航栏实例。

CSS 图片库

CSS 可用来创建图片库。

演示：CSS 图片库


在此处添加对图像的描述

在此处添加对图像的描述

在此处添加对图像的描述

在此处添加对图像的描述

图片库

下面的图片库是由 CSS 创建的：

实例

```
<!doctype html>
<html>
<head>
<style>
div.img
  {
  margin:3px;
  border:1px solid #bebebe;
  height:auto;
  width:auto;
  float:left;
  text-align:center;
  }
div.img img
```

```
  {
  display:inline;
  margin:3px;
  border:1px solid #bebebe;
  }
div.img a:hover img
  {
  border:1px solid #333333;
  }
div.desc
  {
  text-align:center;
  font-weight:normal;
  width:150px;
  font-size:12px;
  margin:10px 5px 10px 5px;
  }
</style>
</head>
<body>

<div class="img">
  <a target="_blank" href="/i/tulip_ballade.jpg">
  <img src="/i/tulip_ballade_s.jpg" alt="Ballade" width="160" heigh
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

<div class="img">
  <a target="_blank" href="/i/tulip_flaming_club.jpg">
  <img src="/i/tulip_flaming_club_s.jpg" alt="Ballade" width="160"
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

<div class="img">
  <a target="_blank" href="/i/tulip_fringed_family.jpg">
  <img src="/i/tulip_fringed_family_s.jpg" alt="Ballade" width="160
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

<div class="img">
  <a target="_blank" href="/i/tulip_peach_blossom.jpg">
  <img src="/i/tulip_peach_blossom_s.jpg" alt="Ballade" width="160"
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

</body>
</html>
```

CSS 图像透明度

通过 CSS 创建透明图像是很容易的。

注释：CSS opacity 属性是 W3C CSS 推荐标准的一部分。

亲自试一试 - 实例

创建透明图像 - Hover 效果

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;style&gt;
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
img:hover
{
opacity:1.0;
filter:alpha(opacity=100); /* For IE8 and earlier */
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;h1&gt;图像透明度&lt;/h1&gt;

&lt;img src="/i/tulip_peach_blossom_w_s.jpg" alt="Peach Blossom" /&

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;在 IE 中，必须添加 &lt;!DOCTYPE&gt;,
&lt;/body&gt;
&lt;/html&gt;
```

创建文本在背景图像上的透明框

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;style&gt;
div.background
{
  width: 400px;
```

```
  height: 266px;
  margin:15px;
  background: url('/i/tulip_peach_blossom_w.jpg') no-repeat;
  border: 1px solid black;
}

div.transbox
{
  width: 338px;
  height: 204px;
  margin:30px;
  padding:0;
  background-color: #ffffff;
  border: 1px solid black;
  /* for IE */
  filter:alpha(opacity=60);
  /* CSS3 standard */
  opacity:0.6;
}

div.transbox p
{
  margin: 30px 40px;
  color: #000000;
  font:bold 12px Verdana, Geneva, sans-serif;
  line-height:1.5;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;div class="background"&gt;
&lt;div class="transbox"&gt;
&lt;p&gt;
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
&lt;/p&gt;
&lt;/div&gt;
&lt;/div&gt;

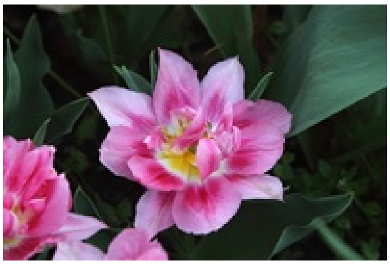
&lt;/body&gt;
&lt;/html&gt;
```

实例 1 - 创建透明图像

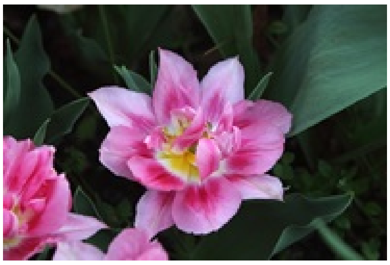
定义透明效果的 CSS3 属性是 *opacity*。

首先，我们将展示如何通过 CSS 来创建透明图像。

常规图像：



带有透明度的相同图像：



请看下面的 CSS：

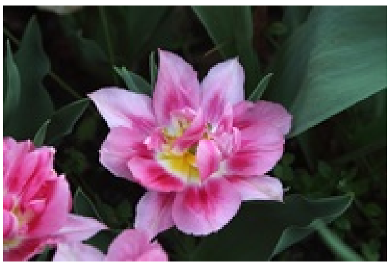
```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* 针对 IE8 以及更早的版本 */
}
```

IE9, Firefox, Chrome, Opera 和 Safari 使用属性 *opacity* 来设定透明度。opacity 属性能够设置的值从 0.0 到 1.0。值越小，越透明。

IE8 以及更早的版本使用滤镜 *filter:alpha(opacity=x)*。x 能够取的值从 0 到 100。值越小，越透明。

实例 2 - 图像透明度 - Hover 效果

请把鼠标指针移动到图像上：



CSS 是这样的：

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* 针对 IE8 以及更早的版本 */
}
img:hover
{
opacity:1.0;
filter:alpha(opacity=100); /* 针对 IE8 以及更早的版本 */
}
```

第一个 CSS 代码块类似实例 1 中的代码。此外，我们已经设置了当鼠标指针位于图像上时的样式。在这个例子中，当指针移动到图像上时，我们希望图像是不透明的。

对应的 CSS 是：*opacity=1*。

IE8 以及更早的浏览器：*filter:alpha(opacity=100)*。

当鼠标指针移出图像后，图像会再次透明。

实例 3 - 透明框中的文本

This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box.

源代码是这样的：

```
<!DOCTYPE html>
<html>
<head>
<style>
div.background
{
  width: 400px;
  height: 266px;
  background: url('/i/tulip_peach_blossom_w.jpg') no-repeat;
  border: 1px solid black;
}

div.transbox
{
  width: 338px;
  height: 204px;
  margin:30px;
  background-color: #ffffff;
  border: 1px solid black;
  /* for IE */
  filter:alpha(opacity=60);
  /* CSS3 standard */
  opacity:0.6;
}

div.transbox p
{
  margin: 30px 40px;
}
</style>
</head>

<body>

<div class="background">
<div class="transbox">
<p>
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>
</div>

</body>
</html>
```

首先，我们创建一个 div 元素 (class="background")，它有固定的高度和宽度、背景图像，以及边框。然后我们在第一个 div 内创建稍小的 div (class="transbox")。"transbox" div 有固定的宽度、背景色和边框 - 并且它是透明的。在透明 div 内部，我们在 p 元素中加入了一些文本。

相关页面

CSS 参考手册：CSS3 opacity 属性

CSS2 媒介类型

媒介类型**(Media Types)**允许你定义以何种媒介来提交文档。文档可以被显示在显示器、纸媒介或者听觉浏览器等等。

媒介类型

某些 CSS 属性仅仅被设计为针对某些媒介。比方说 "voice-family" 属性被设计为针对听觉用户终端。其他的属性可被用于不同的媒介。例如，"font-size" 属性可被用于显示器以及印刷媒介，但是也许会带有不同的值。显示器上面的显示的文档通常会需要比纸媒介文档更大的字号，同时，在显示器上，sans-serif 字体更易阅读，而在纸媒介上，serif 字体更易阅读。

@media规则

@media 规则使你有能力在相同的样式表中，使用不同的样式规则来针对不同的媒介。

下面这个例子中的样式告知浏览器在显示器上显示 14 像素的 Verdana 字体。但是假如页面需要被打印，将使用 10 个像素的 Times 字体。注意：font-weight 被设置为粗体，不论显示器还是纸媒介：

```
<html>
<head>

<style>
@media screen
{
p.test {font-family:verdana,sans-serif; font-size:14px}
}

@media print
{
p.test {font-family:times,serif; font-size:10px}
}

@media screen,print
{
p.test {font-weight:bold}
}
</style>

</head>

<body>....</body>

</html>
```

不同的媒介类型

注释：媒介类型名称对大小写不敏感。

媒介类型	描述
all	用于所有的媒介设备。
aural	用于语音和音频合成器。
braille	用于盲人用点字法触觉回馈设备。
embossed	用于分页的盲人用点字法打印机。
handheld	用于小的手持的设备。
print	用于打印机。
projection	用于方案展示，比如幻灯片。
screen	用于电脑显示器。
tty	用于使用固定密度字母栅格的媒介，比如电传打字机和终端。
tv	用于电视机类型的设备。

CSS 注意事项

本节列出了在使用 **CSS** 时尽量避免使用的技术。

Internet Explorer Behaviors

_它是什么？_Internet Explorer 5 引入了行为 (behaviors)。behaviors 是一种通过使用 CSS 向 HTML 元素添加行为的方法。

为什么要避免它？只有 Internet Explorer 支持 behavior 属性。

用什么代替？请使用 JavaScript 和 HTML DOM 取而代之。

例子 1 - Mouseover Highlight

下面的 HTML 文件中有一个 <style> 元素，它为 <h1> 元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
h1 { behavior: url(behave.htc) }
</style>
</head>

<body>
<h1>Mouse over me!!!</h1>
</body>
</html>
```

下面是 XML 文档 "behave.htc"：

```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

behavior 文件包含了针对元素的 JavaScript 和 事件句柄。

如果您使用 Internet Explorer，可以亲自试一下（把鼠标放在例子中的文本上）。

例子 2 - Typewriter Simulation

下面的 HTML 文件中有一个 <style> 元素，它为 id 为 "typing" 的元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
#typing
{
behavior:url(behave_typing.htc);
font-family:'courier new';
}
</style>
</head>

<body>
<span id="typing" speed="100">IE5 introduced DHTML behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>
</span>
</body>
</html>
```

下面是 XML 文档 "behave.htc"：

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
  {
  clearInterval(t);
  }
}
</script>
```

如果您使用 Internet Explorer，可以亲自试一下。

CSS 高级

CSS3 简介

CSS3 完全向后兼容，因此您不必改变现有的设计。浏览器通常支持 CSS2。

CSS3 模块

CSS3 被划分为模块。

其中最重要的 CSS3 模块包括：

- 选择器
- 框模型
- 背景和边框
- 文本效果
- 2D/3D 转换
- 动画
- 多列布局
- 用户界面

CSS3 标准

W3C 仍然在对 CSS3 规范进行开发。

不过，现代浏览器已经实现了相当多的 CSS3 属性。

CSS3 边框

CSS3 边框

通过 CSS3，您能够创建圆角边框，向矩形添加阴影，使用图片来绘制边框 - 并且不需使用设计软件，比如 PhotoShop。

在本章中，您将学到以下边框属性：

- border-radius
- box-shadow
- border-image

浏览器支持

属性	浏览器支持
border-radius	
box-shadow	
border-image	

Internet Explorer 9+ 支持 border-radius 和 box-shadow 属性。

Firefox、Chrome 以及 Safari 支持所有新的边框属性。

注释：对于 border-image，Safari 5 以及更老的版本需要前缀 -webkit-。

Opera 支持 border-radius 和 box-shadow 属性，但是对于 border-image 需要前缀 -o-。

CSS3 圆角边框

在 CSS2 中添加圆角矩形需要技巧。我们必须为每个圆角使用不同的图片。

在 CSS3 中，创建圆角是非常容易的。

在 CSS3 中，border-radius 属性用于创建圆角：

这个矩形有圆角哦！

实例

向 div 元素添加圆角：

```
div
{
border:2px solid;
border-radius:25px;
-moz-border-radius:25px; /* Old Firefox */
}
```

CSS3 边框阴影

在 CSS3 中，box-shadow 用于向方框添加阴影：

实例

向 div 元素添加方框阴影：

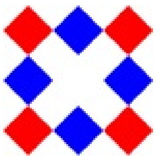
```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

CSS3 边框图片

通过 CSS3 的 border-image 属性，您可以使用图片来创建边框：

border-image 属性允许您规定用于边框的图片！

用于创建上面的边框的原始图片：



实例

使用图片创建围绕 div 元素的边框：

```
div
{
border-image:url(border.png) 30 30 round;
-moz-border-image:url(border.png) 30 30 round; /* 老的 Firefox */
-webkit-border-image:url(border.png) 30 30 round; /* Safari 和 Chro
-o-border-image:url(border.png) 30 30 round; /* Opera */
}
```

新的边框属性

属性	描述	CSS
border-image	设置所有 border-image-* 属性的简写属性。	3
border-radius	设置所有四个 border-*-radius 属性的简写属性。	3
box-shadow	向方框添加一个或多个阴影。	3

CSS3 背景

CSS3 背景

CSS3 包含多个新的背景属性，它们提供了对背景更强大的控制。

在本章，您将学到以下背景属性：

- background-size
- background-origin

您也将学到如何使用多重背景图片。

浏览器支持

属性	浏览器支持
background-size	
background-origin	

Internet Explorer 9+、Firefox、Chrome、Safari 以及 Opera 支持新的背景属性。

CSS3 background-size 属性

background-size 属性规定背景图片的尺寸。

在 CSS3 之前，背景图片的尺寸是由图片的实际尺寸决定的。在 CSS3 中，可以规定背景图片的尺寸，这就允许我们在不同的环境中重复使用背景图片。

您能够以像素或百分比规定尺寸。如果以百分比规定尺寸，那么尺寸相对于父元素的宽度和高度。

例子 1

调整背景图片的大小：

```
div
{
background:url(bg_flower.gif);
-moz-background-size:63px 100px; /* 老版本的 Firefox */
background-size:63px 100px;
background-repeat:no-repeat;
}
```

例子 2

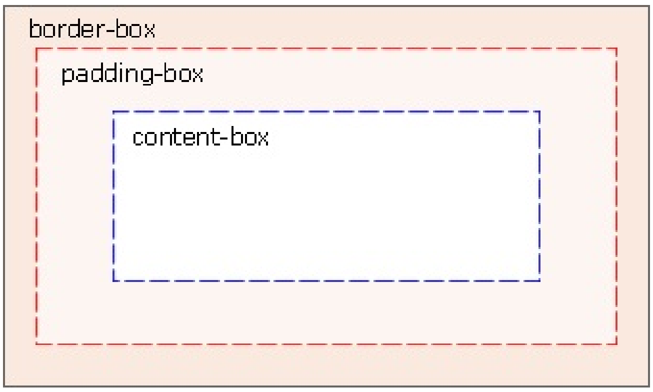
对背景图片进行拉伸，使其完成填充内容区域：

```
div
{
background:url(bg_flower.gif);
-moz-background-size:40% 100%; /* 老版本的 Firefox */
background-size:40% 100%;
background-repeat:no-repeat;
}
```

CSS3 background-origin 属性

background-origin 属性规定背景图片的定位区域。

背景图片可以放置于 content-box、padding-box 或 border-box 区域。



实例

在 content-box 中定位背景图片：

```
div
{
background:url(bg_flower.gif);
background-repeat:no-repeat;
background-size:100% 100%;
-webkit-background-origin:content-box; /* Safari */
background-origin:content-box;
}
```

CSS3 多重背景图片

CSS3 允许您为元素使用多个背景图像。

实例

为 body 元素设置两幅背景图片：

```
body
{
background-image:url(bg_flower.gif),url(bg_flower_2.gif);
}
```

新的背景属性

属性	描述	CSS
background-clip	规定背景的绘制区域。	3
background-origin	规定背景图片的定位区域。	3
background-size	规定背景图片的尺寸。	3

CSS3 文本效果

CSS3 文本效果

CSS3 包含多个新的文本特性。

在本章中，您将学到如下文本属性：

- text-shadow
- word-wrap

浏览器支持

属性	浏览器支持
text-shadow	
word-wrap	

Internet Explorer 10、Firefox、Chrome、Safari 以及 Opera 支持 text-shadow 属性。

所有主流浏览器都支持 word-wrap 属性。

注释：Internet Explorer 9 以及更早的版本，不支持 text-shadow 属性。

CSS3 文本阴影

在 CSS3 中，text-shadow 可向文本应用阴影。

文本阴影效果！

您能够规定水平阴影、垂直阴影、模糊距离，以及阴影的颜色：

实例

向标题添加阴影：

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

CSS3 自动换行

单词太长的话就可能无法超出某个区域：

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

在 CSS3 中，word-wrap 属性允许您允许文本强制文本进行换行 - 即使这意味着会对单词进行拆分：

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

下面是 CSS 代码：

实例

允许对长单词进行拆分，并换行到下一行：

```
p {word-wrap:break-word;}
```

新的文本属性

属性	描述	CSS
hanging-punctuation	规定标点字符是否位于线框之外。	3
punctuation-trim	规定是否对标点字符进行修剪。	3
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行。	3
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色。	3
text-justify	规定当 text-align 设置为 "justify" 时所使用的对齐方法。	3
text-outline	规定文本的轮廓。	3
text-overflow	规定当文本溢出包含元素时发生的事情。	3
text-shadow	向文本添加阴影。	3
text-wrap	规定文本的换行规则。	3
word-break	规定非中日韩文本的换行规则。	3
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行。	3

CSS3 字体

通过 CSS3，Web 设计师再也不必被迫使用“web-safe”字体了。

CSS3 @font-face 规则

在 CSS3 之前，web 设计师必须使用已在用户计算机上安装好的字体。

通过 CSS3，web 设计师可以使用他们喜欢的任意字体。

当您您找到或购买到希望使用的字体时，可将该字体文件存放到 web 服务器上，它会在需要时被自动下载到用户的计算机上。

您“自己的”的字体是在 CSS3 @font-face 规则中定义的。

浏览器支持

属性	浏览器支持
@font-face	

Firefox、Chrome、Safari 以及 Opera 支持 .ttf (True Type Fonts) 和 .otf (OpenType Fonts) 类型的字体。

Internet Explorer 9+ 支持新的 @font-face 规则，但是仅支持 .eot 类型的字体 (Embedded OpenType)。

注释：Internet Explorer 8 以及更早的版本不支持新的 @font-face 规则。

使用您需要的字体

在新的 @font-face 规则中，您必须首先定义字体的名称（比如 myFirstFont），然后指向该字体文件。

如需为 HTML 元素使用字体，请通过 font-family 属性来引用字体的名称 (myFirstFont)：

实例

```
<style>
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Light.ttf'),
     url('Sansation_Light.eot'); /* IE9+ */
}

div
{
font-family:myFirstFont;
}
</style>
```

使用粗体字体

您必须为粗体文本添加另一个包含描述符的 @font-face：

实例

```
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Bold.ttf'),
     url('Sansation_Bold.eot'); /* IE9+ */
font-weight:bold;
}
```

文件 "Sansation_Bold.ttf" 是另一个字体文件，它包含了 Sansation 字体的粗体字符。

只要 font-family 为 "myFirstFont" 的文本需要显示为粗体，浏览器就会使用该字体。

通过这种方式，我们可以为相同的字体设置许多 @font-face 规则。

CSS3 字体描述符

下面的表格列出了能够在 @font-face 规则中定义的所有字体描述符：

描述符	值	描述
font-family	*name*	必需。规定字体的名称。
src	*URL*	必需。定义字体文件的 URL。
font-stretch	normal condensed ultra-condensed extra-condensed semi-condensed expanded semi-expanded extra-expanded ultra-expanded	可选。定义如何拉伸字体。默认是 "normal"。
font-style	ormal italic oblique	可选。定义字体的样式。默认是 "normal"。
font-weight	normal bold 100 200 300 400 500 600 700 800 900	可选。定义字体的粗细。默认是 "normal"。
unicode-range	*unicode-range*	可选。定义字体支持的 UNICODE 字符范围。默认是 "U+0-10FFFF"。

CSS3 2D 转换

CSS3 转换

通过 CSS3 转换，我们能够对元素进行移动、缩放、转动、拉长或拉伸。

它如何工作？

转换是使元素改变形状、尺寸和位置的一种效果。

您可以使用 2D 或 3D 转换来转换您的元素。

浏览器支持

属性	浏览器支持
transform	

Internet Explorer 10、Firefox 以及 Opera 支持 transform 属性。

Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 需要前缀 -ms-。

2D 转换

在本章中，您将学到如下 2D 转换方法：

- translate()
- rotate()
- scale()
- skew()
- matrix()

您将在下一章学习 3D 转换。

实例

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg);        /* IE 9 */
-webkit-transform: rotate(30deg);    /* Safari and Chrome */
-o-transform: rotate(30deg);        /* Opera */
-moz-transform: rotate(30deg);        /* Firefox */
}
```

translate() 方法

通过 translate() 方法，元素从其当前位置移动，根据给定的 left（x 坐标） 和 top（y 坐标） 位置参数：

实例

```
div
{
transform: translate(50px,100px);
-ms-transform: translate(50px,100px);        /* IE 9 */
-webkit-transform: translate(50px,100px);    /* Safari and Chrome *
-o-transform: translate(50px,100px);        /* Opera */
-moz-transform: translate(50px,100px);        /* Firefox */
}
```

值 translate(50px,100px) 把元素从左侧移动 50 像素，从顶端移动 100 像素。

rotate() 方法

通过 rotate() 方法，元素顺时针旋转给定的角度。允许负值，元素将逆时针旋转。

实例

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg);        /* IE 9 */
-webkit-transform: rotate(30deg);    /* Safari and Chrome */
-o-transform: rotate(30deg);        /* Opera */
-moz-transform: rotate(30deg);        /* Firefox */
}
```

值 rotate(30deg) 把元素顺时针旋转 30 度。

scale() 方法

通过 scale() 方法，元素的尺寸会增加或减少，根据给定的宽度（X 轴）和高度（Y 轴）参数：

实例

```
div
{
transform: scale(2,4);
-ms-transform: scale(2,4);    /* IE 9 */
-webkit-transform: scale(2,4);    /* Safari 和 Chrome */
-o-transform: scale(2,4);    /* Opera */
-moz-transform: scale(2,4);    /* Firefox */
}
```

值 scale(2,4) 把宽度转换为原始尺寸的 2 倍，把高度转换为原始高度的 4 倍。

skew() 方法

通过 skew() 方法，元素翻转给定的角度，根据给定的水平线（X 轴）和垂直线（Y 轴）参数：

实例

```
div
{
transform: skew(30deg,20deg);
-ms-transform: skew(30deg,20deg);    /* IE 9 */
-webkit-transform: skew(30deg,20deg);    /* Safari and Chrome */
-o-transform: skew(30deg,20deg);    /* Opera */
-moz-transform: skew(30deg,20deg);    /* Firefox */
}
```

值 skew(30deg,20deg) 围绕 X 轴把元素翻转 30 度，围绕 Y 轴翻转 20 度。

matrix() 方法

matrix() 方法把所有 2D 转换方法组合在一起。

matrix() 方法需要六个参数，包含数学函数，允许您：旋转、缩放、移动以及倾斜元素。

实例

如何使用 matrix 方法将 div 元素旋转 30 度：

```
div
{
transform:matrix(0.866,0.5,-0.5,0.866,0,0);
-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0);		/* IE 9 */
-moz-transform:matrix(0.866,0.5,-0.5,0.866,0,0);	/* Firefox */
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0);	/* Safari ar
-o-transform:matrix(0.866,0.5,-0.5,0.866,0,0);		/* Opera */
}
```

新的转换属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3

2D Transform 方法

函数	描述
matrix(*n*,*n*,*n*,*n*,*n*,*n*)	定义 2D 转换，使用六个值的矩阵。
translate(*x*,*y*)	定义 2D 转换，沿着 X 和 Y 轴移动元素。
translateX(*n*)	定义 2D 转换，沿着 X 轴移动元素。
translateY(*n*)	定义 2D 转换，沿着 Y 轴移动元素。
scale(*x*,*y*)	定义 2D 缩放转换，改变元素的宽度和高度。
scaleX(*n*)	定义 2D 缩放转换，改变元素的宽度。
scaleY(*n*)	定义 2D 缩放转换，改变元素的高度。
rotate(*angle*)	定义 2D 旋转，在参数中规定角度。
skew(*x-angle*,*y-angle*)	定义 2D 倾斜转换，沿着 X 和 Y 轴。
skewX(*angle*)	定义 2D 倾斜转换，沿着 X 轴。
skewY(*angle*)	定义 2D 倾斜转换，沿着 Y 轴。

CSS3 3D 转换

3D 转换

CSS3 允许您使用 3D 转换来对元素进行格式化。

在本章中，您将学到其中的一些 3D 转换方法：

- rotateX()
- rotateY()

点击下面的元素，来查看 2D 转换与 3D 转换之间的不同之处：

2D 旋转

3D 旋转

它如何工作？

转换是使元素改变形状、尺寸和位置的一种效果。

您可以使用 2D 或 3D 转换来转换您的元素。

浏览器支持

属性	浏览器支持
transform	

Internet Explorer 10 和 Firefox 支持 3D 转换。

Chrome 和 Safari 需要前缀 -webkit-。

Opera 仍然不支持 3D 转换（它只支持 2D 转换）。

rotateX() 方法

通过 rotateX() 方法，元素围绕其 X 轴以给定的度数进行旋转。

实例

```
div
{
transform: rotateX(120deg);
-webkit-transform: rotateX(120deg);    /* Safari 和 Chrome */
-moz-transform: rotateX(120deg);    /* Firefox */
}
```

rotateY() 旋转

通过 rotateY() 方法，元素围绕其 Y 轴以给定的度数进行旋转。

实例

```
div
{
transform: rotateY(130deg);
-webkit-transform: rotateY(130deg);    /* Safari 和 Chrome */
-moz-transform: rotateY(130deg);    /* Firefox */
}
```

转换属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3
transform-style	规定被嵌套元素如何在 3D 空间中显示。	3
perspective	规定 3D 元素的透视效果。	3
perspective-origin	规定 3D 元素的底部位置。	3
backface-visibility	定义元素在不面对屏幕时是否可见。	3

2D Transform 方法

函数	描述
matrix3d(*n*,*n*,*n*,*n*,*n*,*n*, *n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*)	定义 3D 转换，使用 16 个值的 4x4 矩阵。
translate3d(*x*,*y*,*z*)	定义 3D 转化。
translateX(*x*)	定义 3D 转化，仅使用用于 X 轴的值。
translateY(*y*)	定义 3D 转化，仅使用用于 Y 轴的值。
translateZ(*z*)	定义 3D 转化，仅使用用于 Z 轴的值。
scale3d(*x*,*y*,*z*)	定义 3D 缩放转换。
scaleX(*x*)	定义 3D 缩放转换，通过给定一个 X 轴的值。
scaleY(*y*)	定义 3D 缩放转换，通过给定一个 Y 轴的值。
scaleZ(*z*)	定义 3D 缩放转换，通过给定一个 Z 轴的值。
rotate3d(*x*,*y*,*z*,*angle*)	定义 3D 旋转。
rotateX(*angle*)	定义沿 X 轴的 3D 旋转。
rotateY(*angle*)	定义沿 Y 轴的 3D 旋转。
rotateZ(*angle*)	定义沿 Z 轴的 3D 旋转。
perspective(*n*)	定义 3D 转换元素的透视视图。

CSS3 过渡

CSS3 过渡

通过 CSS3，我们可以在不使用 Flash 动画或 JavaScript 的情况下，当元素从一种样式变换为另一种样式时为元素添加效果。

请把鼠标移动到下面的元素上：

CSS3 过渡

浏览器支持

属性	浏览器支持
transition	

Internet Explorer 10、Firefox、Chrome 以及 Opera 支持 transition 属性。

Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 以及更早的版本，不支持 transition 属性。

注释：Chrome 25 以及更早的版本，需要前缀 -webkit-。

它如何工作？

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果。

要实现这一点，必须规定两项内容：

- 规定您希望把效果添加到哪个 CSS 属性上
- 规定效果的时长

实例

应用于宽度属性的过渡效果，时长为 2 秒：

```
div
{
transition: width 2s;
-moz-transition: width 2s;    /* Firefox 4 */
-webkit-transition: width 2s;    /* Safari 和 Chrome */
-o-transition: width 2s;    /* Opera */
}
```

注释：如果时长未规定，则不会有过渡效果，因为默认值是 0。

效果开始于指定的 CSS 属性改变值时。CSS 属性改变的典型时间是鼠标指针位于元素上时：

实例

规定当鼠标指针悬浮于 <div> 元素上时：

```
div:hover
{
width:300px;
}
```

注释：当指针移出元素时，它会逐渐变回原来的样式。

多项改变

如需向多个样式添加过渡效果，请添加多个属性，由逗号隔开：

实例

向宽度、高度和转换添加过渡效果：

```
div
{
transition: width 2s, height 2s, transform 2s;
-moz-transition: width 2s, height 2s, -moz-transform 2s;
-webkit-transition: width 2s, height 2s, -webkit-transform 2s;
-o-transition: width 2s, height 2s,-o-transform 2s;
}
```

过渡属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
transition	简写属性，用于在一个属性中设置四个过渡属性。	3
transition-property	规定应用过渡的 CSS 属性的名称。	3
transition-duration	定义过渡效果花费的时间。默认是 0。	3
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。	3
transition-delay	规定过渡效果何时开始。默认是 0。	3

下面的两个例子设置所有过渡属性：

实例

在一个例子中使用所有过渡属性：

```
div
{
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
/* Firefox 4 */
-moz-transition-property:width;
-moz-transition-duration:1s;
-moz-transition-timing-function:linear;
-moz-transition-delay:2s;
/* Safari 和 Chrome */
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
/* Opera */
-o-transition-property:width;
-o-transition-duration:1s;
-o-transition-timing-function:linear;
-o-transition-delay:2s;
}
```

实例

与上面的例子相同的过渡效果，但是使用了简写的 transition 属性：

```
div
{
transition: width 1s linear 2s;
/* Firefox 4 */
-moz-transition:width 1s linear 2s;
/* Safari and Chrome */
-webkit-transition:width 1s linear 2s;
/* Opera */
-o-transition:width 1s linear 2s;
}
```

CSS3 动画

CSS3 动画

通过 CSS3，我们能够创建动画，这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript。

CSS3 动画

CSS3 @keyframes 规则

如需在 CSS3 中创建动画，您需要学习 @keyframes 规则。

@keyframes 规则用于创建动画。在 @keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。

浏览器支持

属性	浏览器支持
@keyframes	
animation	

Internet Explorer 10、Firefox 以及 Opera 支持 @keyframes 规则和 animation 属性。

Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9，以及更早的版本，不支持 @keyframe 规则或 animation 属性。

实例

```
@keyframes myfirst
{
from {background: red;}
to {background: yellow;}
}

@-moz-keyframes myfirst /* Firefox */
{
from {background: red;}
to {background: yellow;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
from {background: red;}
to {background: yellow;}
}

@-o-keyframes myfirst /* Opera */
{
from {background: red;}
to {background: yellow;}
}
```

CSS3 动画

当您在 @keyframes 中创建动画时，请把它捆绑到某个选择器，否则不会产生动画效果。

通过规定至少以下两项 CSS3 动画属性，即可将动画绑定到选择器：

- 规定动画的名称
- 规定动画的时长

实例

把 "myfirst" 动画捆绑到 div 元素，时长：5 秒：

```
div
{
animation: myfirst 5s;
-moz-animation: myfirst 5s;	/* Firefox */
-webkit-animation: myfirst 5s;	/* Safari 和 Chrome */
-o-animation: myfirst 5s;	/* Opera */
}
```

注释：您必须定义动画的名称和时长。如果忽略时长，则动画不会允许，因为默认值是 0。

什么是 CSS3 中的动画？

动画是使元素从一种样式逐渐变化为另一种样式的效果。

您可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%。

0% 是动画的开始，100% 是动画的完成。

为了得到最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

实例

当动画为 25% 及 50% 时改变背景色，然后当动画 100% 完成时再次改变：

```
@keyframes myfirst
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-moz-keyframes myfirst /* Firefox */
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-o-keyframes myfirst /* Opera */
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}
```

实例

改变背景色和位置：

```
@keyframes myfirst
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

@-moz-keyframes myfirst /* Firefox */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

@-o-keyframes myfirst /* Opera */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

CSS3 动画属性

下面的表格列出了 @keyframes 规则和所有动画属性：

属性	描述	CSS
@keyframes	规定动画。	3
animation	所有动画属性的简写属性，除了 animation-play-state 属性。	3
animation-name	规定 @keyframes 动画的名称。	3
animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。	3
animation-timing-function	规定动画的速度曲线。默认是 "ease"。	3
animation-delay	规定动画何时开始。默认是 0。	3
animation-iteration-count	规定动画被播放的次数。默认是 1。	3
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。	3
animation-play-state	规定动画是否正在运行或暂停。默认是 "running"。	3
animation-fill-mode	规定对象动画时间之外的状态。	3

下面的两个例子设置了所有动画属性：

实例

运行名为 myfirst 的动画，其中设置了所有动画属性：

```
div
{
animation-name: myfirst;
animation-duration: 5s;
animation-timing-function: linear;
animation-delay: 2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
/* Firefox: */
-moz-animation-name: myfirst;
-moz-animation-duration: 5s;
-moz-animation-timing-function: linear;
-moz-animation-delay: 2s;
-moz-animation-iteration-count: infinite;
-moz-animation-direction: alternate;
-moz-animation-play-state: running;
/* Safari 和 Chrome: */
-webkit-animation-name: myfirst;
-webkit-animation-duration: 5s;
-webkit-animation-timing-function: linear;
-webkit-animation-delay: 2s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-play-state: running;
/* Opera: */
-o-animation-name: myfirst;
-o-animation-duration: 5s;
-o-animation-timing-function: linear;
-o-animation-delay: 2s;
-o-animation-iteration-count: infinite;
-o-animation-direction: alternate;
-o-animation-play-state: running;
}
```

实例

与上面的动画相同，但是使用了简写的动画 animation 属性：

```
div
{
animation: myfirst 5s linear 2s infinite alternate;
/* Firefox: */
-moz-animation: myfirst 5s linear 2s infinite alternate;
/* Safari 和 Chrome: */
-webkit-animation: myfirst 5s linear 2s infinite alternate;
/* Opera: */
-o-animation: myfirst 5s linear 2s infinite alternate;
}
```

CSS3 多列

CSS3 多列

通过 CSS3，您能够创建多个列来对文本进行布局 - 就像报纸那样！

在本章中，您将学习如下多列属性：

- column-count
- column-gap
- column-rule

浏览器支持

属性	浏览器支持
column-count	
column-gap	
column-rule	

Internet Explorer 10 和 Opera 支持多列属性。

Firefox 需要前缀 -moz-。

Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 以及更早的版本不支持多列属性。

CSS3 创建多列

column-count 属性规定元素应该被分隔的列数：

实例

把 div 元素中的文本分隔为三列：

```
div
{
-moz-column-count:3;     /* Firefox */
-webkit-column-count:3; /* Safari 和 Chrome */
column-count:3;
}
```

CSS3 规定列之间的间隔

column-gap 属性规定列之间的间隔：

实例

规定列之间 40 像素的间隔：

```
div
{
-moz-column-gap:40px;        /* Firefox */
-webkit-column-gap:40px;    /* Safari 和 Chrome */
column-gap:40px;
}
```

CSS3 列规则

column-rule 属性设置列之间的宽度、样式和颜色规则。

实例

规定列之间的宽度、样式和颜色规则：

```
div
{
-moz-column-rule:3px outset #ff0000;    /* Firefox */
-webkit-column-rule:3px outset #ff0000;    /* Safari and Chrome */
column-rule:3px outset #ff0000;
}
```

新的多列属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
column-count	规定元素应该被分隔的列数。	3
column-fill	规定如何填充列。	3
column-gap	规定列之间的间隔。	3
column-rule	设置所有 column-rule-* 属性的简写属性。	3
column-rule-color	规定列之间规则的颜色。	3
column-rule-style	规定列之间规则的样式。	3
column-rule-width	规定列之间规则的宽度。	3
column-span	规定元素应该横跨的列数。	3
column-width	规定列的宽度。	3
columns	规定设置 column-width 和 column-count 的简写属性。	3

CSS3 用户界面

CSS3 用户界面

在 CSS3 中，新的用户界面特性包括重设元素尺寸、盒尺寸以及轮廓等。

在本章中，您将学到以下用户界面属性：

- resize
- box-sizing
- outline-offset

浏览器支持

属性	浏览器支持
resize	
box-sizing	
outline-offset	

Firefox、Chrome 以及 Safari 支持 resize 属性。

Internet Explorer、Chrome、Safari 以及 Opera 支持 box-sizing 属性。Firefox 需要前缀 -moz-。

所有主流浏览器都支持 outline-offset 属性，除了 Internet Explorer。

CSS3 Resizing

在 CSS3，resize 属性规定是否可由用户调整元素尺寸。

这个 div 元素可由用户调整尺寸（在 Firefox 4+、Chrome 以及 Safari 中）。

CSS 代码如下：

实例

规定 div 元素可由用户调整大小：

```
div
{
resize:both;
overflow:auto;
}
```

CSS3 Box Sizing

box-sizing 属性允许您以确切的方式定义适应某个区域的具体内容。

实例

规定两个并排的带边框方框：

```
div
{
box-sizing:border-box;
-moz-box-sizing:border-box;    /* Firefox */
-webkit-box-sizing:border-box;    /* Safari */
width:50%;
float:left;
}
```

CSS3 Outline Offset

outline-offset 属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

轮廓与边框有两点不同：

- 轮廓不占用空间
- 轮廓可能是非矩形

这个 div 在边框之外 15 像素处有一个轮廓。

CSS 代码如下：

实例

规定边框边缘之外 15 像素处的轮廓：

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

新的用户界面属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
appearance	允许您将元素设置为标准用户界面元素的外观	3
box-sizing	允许您以确切的方式定义适应某个区域的具体内容。	3
icon	为创作者提供使用图标化等价物来设置元素样式的能力。	3
nav-down	规定在使用 arrow-down 导航键时向何处导航。	3
nav-index	设置元素的 tab 键控制次序。	3
nav-left	规定在使用 arrow-left 导航键时向何处导航。	3
nav-right	规定在使用 arrow-right 导航键时向何处导航。	3
nav-up	规定在使用 arrow-up 导航键时向何处导航。	3
outline-offset	对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。	3
resize	规定是否可由用户对元素的尺寸进行调整。	3

W3School CSS 参考手册

来源：CSS 参考手册

整理：飞龙

CSS3 动画属性（Animation）

属性	描述	CSS
@keyframes	规定动画。	3
animation	所有动画属性的简写属性，除了 animation-play-state 属性。	3
animation-name	规定 @keyframes 动画的名称。	3
animation-duration	规定动画完成一个周期所花费的秒或毫秒。	3
animation-timing-function	规定动画的速度曲线。	3
animation-delay	规定动画何时开始。	3
animation-iteration-count	规定动画被播放的次数。	3
animation-direction	规定动画是否在下一周期逆向地播放。	3
animation-play-state	规定动画是否正在运行或暂停。	3
animation-fill-mode	规定对象动画时间之外的状态。	3

CSS3 @keyframes 规则

实例

使 div 元素匀速向下移动：

```
@keyframes mymove
{
from {top:0px;}
to {top:200px;}
}

@-moz-keyframes mymove /* Firefox */
{
from {top:0px;}
to {top:200px;}
}

@-webkit-keyframes mymove /* Safari 和 Chrome */
{
from {top:0px;}
to {top:200px;}
}

@-o-keyframes mymove /* Opera */
{
from {top:0px;}
to {top:200px;}
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前浏览器都不支持 @keyframes 规则。

Firefox 支持替代的 @-moz-keyframes 规则。

Opera 支持替代的 @-o-keyframes 规则。

Safari 和 Chrome 支持替代的 @-webkit-keyframes 规则。

定义和用法

通过 @keyframes 规则，您能够创建动画。

创建动画的原理是，将一套 CSS 样式逐渐变化为另一套样式。

在动画过程中，您能够多次改变这套 CSS 样式。

以百分比来规定改变发生的时间，或者通过关键词 "from" 和 "to"，等价于 0% 和 100%。

0% 是动画的开始时间，100% 动画的结束时间。

为了获得最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

注释：请使用动画属性来控制动画的外观，同时将动画与选择器绑定。

语法

```
@keyframes _animationname_ {_keyframes-selector_ {_css-styles_;}}
```

值	描述
animationname	必需。定义动画的名称。
keyframes-selector	

必需。动画时长的百分比。

合法的值：

- 0-100%
- from（与 0% 相同）
- to（与 100% 相同）

  | | *css-styles* | 必需。一个或多个合法的 CSS 样式属性。 |

亲自试一试 - 实例

实例 1

在一个动画中添加多个 keyframe 选择器：

```
@keyframes mymove
{
0%   {top:0px;}
25%  {top:200px;}
50%  {top:100px;}
75%  {top:200px;}
100% {top:0px;}
}

@-moz-keyframes mymove /* Firefox */
{
0%   {top:0px;}
25%  {top:200px;}
50%  {top:100px;}
75%  {top:200px;}
100% {top:0px;}
}

@-webkit-keyframes mymove /* Safari 和 Chrome */
{
0%   {top:0px;}
25%  {top:200px;}
50%  {top:100px;}
75%  {top:200px;}
100% {top:0px;}
}

@-o-keyframes mymove /* Opera */
{
0%   {top:0px;}
25%  {top:200px;}
50%  {top:100px;}
75%  {top:200px;}
100% {top:0px;}
}
```

实例 2

在一个动画中改变多个 CSS 样式：

```
@keyframes mymove
{
0%   {top:0px; background:red; width:100px;}
100% {top:200px; background:yellow; width:300px;}
}

@-moz-keyframes mymove /* Firefox */
{
0%   {top:0px; background:red; width:100px;}
100% {top:200px; background:yellow; width:300px;}
}

@-webkit-keyframes mymove /* Safari 和 Chrome */
{
0%   {top:0px; background:red; width:100px;}
100% {top:200px; background:yellow; width:300px;}
}

@-o-keyframes mymove /* Opera */
{
0%   {top:0px; background:red; width:100px;}
100% {top:200px; background:yellow; width:300px;}
}
```

实例 3

带有多个 CSS 样式的多个 keyframe 选择器：

```
@keyframes mymove
{
0%   {top:0px; left:0px; background:red;}
25%  {top:0px; left:100px; background:blue;}
50%  {top:100px; left:100px; background:yellow;}
75%  {top:100px; left:0px; background:green;}
100% {top:0px; left:0px; background:red;}
}

@-moz-keyframes mymove /* Firefox */
{
0%   {top:0px; left:0px; background:red;}
25%  {top:0px; left:100px; background:blue;}
50%  {top:100px; left:100px; background:yellow;}
75%  {top:100px; left:0px; background:green;}
100% {top:0px; left:0px; background:red;}
}

@-webkit-keyframes mymove /* Safari and Chrome */
{
0%   {top:0px; left:0px; background:red;}
25%  {top:0px; left:100px; background:blue;}
50%  {top:100px; left:100px; background:yellow;}
75%  {top:100px; left:0px; background:green;}
100% {top:0px; left:0px; background:red;}
}

@-o-keyframes mymove /* Opera */
{
0%   {top:0px; left:0px; background:red;}
25%  {top:0px; left:100px; background:blue;}
50%  {top:100px; left:100px; background:yellow;}
75%  {top:100px; left:0px; background:green;}
100% {top:0px; left:0px; background:red;}
}
```

相关页面

CSS3 教程：CSS3 动画

CSS3 animation 属性

实例

使用简写属性，将动画与 div 元素绑定：

```
div
{
animation:mymove 5s infinite;
-webkit-animation:mymove 5s infinite; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation 属性。

Safari 和 Chrome 支持替代的 -webkit-animation 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation 属性。

定义和用法

animation 属性是一个简写属性，用于设置六个动画属性：

- animation-name
- animation-duration
- animation-timing-function
- animation-delay
- animation-iteration-count
- animation-direction

注释：请始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。

默认值：	none 0 ease 0 1 normal
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.animation="mymove 5s infinite"

语法

```
animation: _name_ _duration_ _timing-function_ _delay_ _iteration-c
```

值	描述
animation-name	规定需要绑定到选择器的 keyframe 名称。。
animation-duration	规定完成动画所花费的时间，以秒或毫秒计。
animation-timing-function	规定动画的速度曲线。
animation-delay	规定在动画开始之前的延迟。
animation-iteration-count	规定动画应该播放的次数。
animation-direction	规定是否应该轮流反向播放动画。

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-name 属性

实例

为 @keyframes 动画规定一个名称：

```
div
{
animation:mymove 5s infinite;
-webkit-animation:mymove 5s infinite; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-name 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-name 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-name 属性。

定义和用法

animation-name 属性为 @keyframes 动画规定名称。

注释：请始终规定 animation-duration 属性，否则时长为 0，就不会播放动画了。

默认值：	none
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.animationName="mymove"
---	---

语法

```
animation-name: _keyframename_|none;
```

值	描述
keyframename	规定需要绑定到选择器的 keyframe 的名称。
none	规定无动画效果（可用于覆盖来自级联的动画）。

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-duration 属性

实例

为 @keyframes 动画规定一个名称：

```
div
{
animation-duration:2s;
-webkit-animation-duration:2s; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-duration 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-duration 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-duration 属性。

定义和用法

animation-duration 属性定义动画完成一个周期所需要的时间，以秒或毫秒计。

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.animationDuration="3s"

语法

```
animation-duration: _time_;
```

值	描述
time	规定完成动画所花费的时间。默认值是 0，意味着没有动画效果。

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-timing-function 属性

实例

从开头到结尾以相同的速度来播放动画：

```
div
{
animation-timing-function:2s;
-webkit-animation-timing-function:2s; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-timing-function 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-timing-function 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-timing-function 属性。

定义和用法

animation-timing-function 规定动画的速度曲线。

速度曲线定义动画从一套 CSS 样式变为另一套所用的时间。

速度曲线用于使变化更为平滑。

默认值：	ease
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.animationTimingFunction="linear"

语法

```
animation-timing-function: _value_;
```

animation-timing-function 使用名为三次贝塞尔（Cubic Bezier）函数的数学函数，来生成速度曲线。您能够在该函数中使用自己的值，也可以预定义的值：

值	描述	测试
linear	动画从头到尾的速度是相同的。	测试
ease	默认。动画以低速开始，然后加快，在结束前变慢。	测试
ease-in	动画以低速开始。	测试
ease-out	动画以低速结束。	测试
ease-in-out	动画以低速开始和结束。	测试
cubic-bezier(*n*,*n*,*n*,*n*)	在 cubic-bezier 函数中自己的值。可能的值是从 0 到 1 的数值。	

提示：请试着在下面的“亲自试一试”功能中使用不同的值。

亲自试一试 - 实例

实例 1

为了更好地理解不同的定时函数值，这里提供了设置五个不同值的五个不同的 div 元素：

```
/* W3C 和 Opera: */
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
/* Firefox: */
#div1 {-moz-animation-timing-function: linear;}
#div2 {-moz-animation-timing-function: ease;}
#div3 {-moz-animation-timing-function: ease-in;}
#div4 {-moz-animation-timing-function: ease-out;}
#div5 {-moz-animation-timing-function: ease-in-out;}
/* Safari 和 Chrome: */
#div1 {-webkit-animation-timing-function: linear;}
#div2 {-webkit-animation-timing-function: ease;}
#div3 {-webkit-animation-timing-function: ease-in;}
#div4 {-webkit-animation-timing-function: ease-out;}
#div5 {-webkit-animation-timing-function: ease-in-out;}
```

实例 2

与上例相同，但是通过 cubic-bezier 函数来定义速度曲线：

```
/* W3C 和 Opera: */
#div1 {animation-timing-function: cubic-bezier(0,0,1,1);}
#div2 {animation-timing-function: cubic-bezier(0.25,0.1,0.25,1);}
#div3 {animation-timing-function: cubic-bezier(0.42,0,1,1);}
#div4 {animation-timing-function: cubic-bezier(0,0,0.58,1);}
#div5 {animation-timing-function: cubic-bezier(0.42,0,0.58,1);}
/* Firefox: */
#div1 {-moz-animation-timing-function: cubic-bezier(0,0,1,1);}
#div2 {-moz-animation-timing-function: cubic-bezier(0.25,0.1,0.25,1
#div3 {-moz-animation-timing-function: cubic-bezier(0.42,0,1,1);}
#div4 {-moz-animation-timing-function: cubic-bezier(0,0,0.58,1);}
#div5 {-moz-animation-timing-function: cubic-bezier(0.42,0,0.58,1)
/* Safari 和 Chrome: */
#div1 {-webkit-animation-timing-function: cubic-bezier(0,0,1,1);}
#div2 {-webkit-animation-timing-function: cubic-bezier(0.25,0.1,0.2
#div3 {-webkit-animation-timing-function: cubic-bezier(0.42,0,1,1)
#div4 {-webkit-animation-timing-function: cubic-bezier(0,0,0.58,1)
#div5 {-webkit-animation-timing-function: cubic-bezier(0.42,0,0.58
```

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-delay 属性

实例

等待两秒，然后开始动画：

```
div
{
animation-delay:2s;
-webkit-animation-delay:2s; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-delay 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-delay 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-delay 属性。

定义和用法

animation-delay 属性定义动画何时开始。

animation-delay 值以秒或毫秒计。

提示：允许负值，-2s 使动画马上开始，但跳过 2 秒进入动画。

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.animationDelay="2s"

语法

```
animation-delay: _time_;
```

值	描述	测试
time	可选。定义动画开始前等待的时间，以秒或毫秒计。默认值是 0。	测试

亲自试一试 - 实例

负值，请注意动画跳过 2 秒进入动画周期：

```
animation-delay: -2s /* W3C 和 Opera */
-moz-animation-delay: -2s /* Firefox */
-webkit-animation-delay: -2s /* Safari 和 Chrome */
```

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-iteration-count 属性

实例

播放动画三次：

```
div
{
animation-iteration-count:3;
-webkit-animation-iteration-count:3; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-iteration-count 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-iteration-count 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-iteration-count 属性。

定义和用法

animation-iteration-count 属性定义动画的播放次数。

默认值：	1
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.animationIterationCount=3
---	---

语法

```
animation-iteration-count: _n_|infinite;
```

值	描述	测试
n	定义动画播放次数的数值。	测试
infinite	规定动画应该无限次播放。	测试

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-direction 属性

实例

暂停动画：

```
div
{
animation-direction:alternate;
-webkit-animation-direction:alternate; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-direction 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-direction 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-direction 属性。

定义和用法

animation-direction 属性定义是否应该轮流反向播放动画。

如果 animation-direction 值是 "alternate"，则动画会在奇数次数（1、3、5 等等）正常播放，而在偶数次数（2、4、6 等等）向后播放。

注释：如果把动画设置为只播放一次，则该属性没有效果。

默认值：	**normal**
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.animationDirection="alternate"
---	---

语法

```
animation-direction: normal|alternate;
```

值	描述	测试
normal	默认值。动画应该正常播放。	测试
alternate	动画应该轮流反向播放。	测试

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-play-state 属性

实例

暂停动画：

```
div
{
animation-play-state:paused;
-webkit-animation-play-state:paused; /* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-play-state 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-play-state 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-play-state 属性。

定义和用法

animation-play-state 属性规定动画正在运行还是暂停。

注释：您可以在 JavaScript 中使用该属性，这样就能在播放过程中暂停动画。

默认值：	running
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.animationPlayState="paused"

语法

```
animation-play-state: paused|running;
```

值	描述	测试
paused	规定动画已暂停。	测试
running	规定动画正在播放。	测试

相关页面

CSS3 教程：CSS3 动画

CSS3 animation-fill-mode 属性

实例

为 h1 元素规定填充模式：

```
h1
  {
  animation-fill-mode: forwards;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox 以及 Opera 支持 animation-fill-mode 属性。

Safari 和 Chrome 支持替代的 -webkit-animation-fill-mode 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation-fill-mode 属性。

定义和用法

animation-fill-mode 属性规定动画在播放之前或之后，其动画效果是否可见。

注释：其属性值是由逗号分隔的一个或多个填充模式关键词。

默认值：	none
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.animationFillMode=none

语法

```
animation-fill-mode : none | forwards | backwards | both;
```

值	描述
none	不改变默认行为。
forwards	当动画完成后，保持最后一个属性值（在最后一个关键帧中定义）。
backwards	在 animation-delay 所指定的一段时间内，在动画显示之前，应用开始属性值（在第一个关键帧中定义）。
both	向前和向后填充模式都被应用。

相关页面

CSS3 教程：CSS3 动画

CSS 背景属性（Background）

属性	描述	CSS
background	在一个声明中设置所有的背景属性。	1
background-attachment	设置背景图像是否固定或者随着页面的其余部分滚动。	1
background-color	设置元素的背景颜色。	1
background-image	设置元素的背景图像。	1
background-position	设置背景图像的开始位置。	1
background-repeat	设置是否及如何重复背景图像。	1
background-clip	规定背景的绘制区域。	3
background-origin	规定背景图片的定位区域。	3
background-size	规定背景图片的尺寸。	3

CSS background 属性

实例

如何在一个声明中设置所有背景属性：

```
body
  { 
  background: #00FF00 url(bgimage.gif) no-repeat fixed top;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 background 属性。

注释：IE8 以及更早的浏览器不支持一个元素多个背景图像。

注释：IE7 以及更早的浏览器不支持 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

background 简写属性在一个声明中设置所有的背景属性。

可以设置如下属性：

- background-color
- background-position
- background-size
- background-repeat
- background-origin
- background-clip
- background-attachment
- background-image

如果不设置其中的某个值，也不会出问题，比如 background:#ff0000 url('smiley.gif'); 也是允许的。

通常建议使用这个属性，而不是分别使用单个属性，因为这个属性在较老的浏览器中能够得到更好的支持，而且需要键入的字母也更少。

默认值：	***not specified***
继承性：	no
---	---
版本：	CSS1 + CSS3
---	---
JavaScript 语法：	*object*.style.background="white url(paper.gif) repeat-y"
---	---

可能的值

值	描述	CSS
background-color	规定要使用的背景颜色。	1
background-position	规定背景图像的位置。	1
background-size	规定背景图片的尺寸。	3
background-repeat	规定如何重复背景图像。	1
background-origin	规定背景图片的定位区域。	3
background-clip	规定背景的绘制区域。	3
background-attachment	规定背景图像是否固定或者随着页面的其余部分滚动。	1
background-image	规定要使用的背景图像。	1
inherit	规定应该从父元素继承 background 属性的设置。	1

亲自试一试 - 实例

所有背景属性在一个声明之中

本例演示如何使用简写属性来将所有背景属性设置在一个声明之中。

```
<html>
<head>
<style type="text/css">
body
{
background: #ff0000 url(/i/eg_bg_03.gif) no-repeat fixed center;
}
</style>
</head>

<body>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
<p>这是一些文本。</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 背景

CSS3 教程：CSS3 背景

HTML DOM 参考手册：background 属性

CSS background-attachment 属性

实例

如何设置固定的背景图像：

```
body
  {
  background-image: url(bgimage.gif);
  background-attachment: fixed;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 background-attachment 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

background-attachment 属性设置背景图像是否固定或者随着页面的其余部分滚动。

默认值：	scroll
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.backgroundAttachment="fixed"
---	---

可能的值

值	描述
scroll	默认值。背景图像会随着页面其余部分的滚动而移动。
fixed	当页面的其余部分滚动时，背景图像不会移动。
inherit	规定应该从父元素继承 background-attachment 属性的设置。

TIY 实例

如何设置固定的背景图像

本例演示如何设置固定的背景图像。图像不会随着页面的其他部分滚动。

```
<html>
<head>
<style type="text/css">
body
{
background-image:url(/i/eg_bg_02.gif);
background-repeat:no-repeat;
background-attachment:fixed
}
</style>
</head>

<body>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 背景

HTML DOM 参考手册：backgroundAttachment 属性

CSS background-color 属性

实例

```
body
  {
  background-color:yellow;
  }
h1
  {
  background-color:#00ff00;
  }
p
  {
  background-color:rgb(255,0,255);
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 background-color 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

background-color 属性设置元素的背景颜色。

元素背景的范围

background-color 属性为元素设置一种纯色。这种颜色会填充元素的内容、内边距和边框区域，扩展到元素边框的外边界（但不包括外边距）。如果边框有透明部分（如虚线边框），会透过这些透明部分显示出背景色。

transparent 值

尽管在大多数情况下，没有必要使用 transparent。不过如果您不希望某元素拥有背景色，同时又不希望用户对浏览器的颜色设置影响到您的设计，那么设置 transparent 值还是有必要的。

默认值：	**transparent**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.backgroundColor="#00FF00"

可能的值

值	描述
color_name	规定颜色值为颜色名称的背景颜色（比如 red）。
hex_number	规定颜色值为十六进制值的背景颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的背景颜色（比如 rgb(255,0,0)）。
transparent	默认。背景颜色为透明。
inherit	规定应该从父元素继承 background-color 属性的设置。

TIY 实例

设置背景颜色

本例演示如何为元素设置背景颜色。

```
<html>
<head>

<style type="text/css">

body {background-color: yellow}
h1 {background-color: #00ff00}
h2 {background-color: transparent}
p {background-color: rgb(250,0,255)}

p.no2 {background-color: gray; padding: 20px;}

</style>

</head>

<body>

<h1>这是标题 1</h1>
<h2>这是标题 2</h2>
<p>这是段落</p>
<p class="no2">这个段落设置了内边距。</p>

</body>
</html>
```

设置文本的背景颜色

本例颜色如何设置部分文本的背景颜色。

```
<html>
<head>
<style type="text/css">
span.highlight
{
background-color:yellow
}
</style>
</head>

<body>
<p>
<span class="highlight">这是文本。</span> 这是文本。 这是文本。 这是文本。
</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 背景

HTML DOM 参考手册：backgroundColor 属性

CSS background-image 属性

实例

```
body
  {
  background-image: url(bgimage.gif);
  background-color: #000000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 background-image 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

background-image 属性为元素设置背景图像。

元素的背景占据了元素的全部尺寸，包括内边距和边框，但不包括外边距。

默认地，背景图像位于元素的左上角，并在水平和垂直方向上重复。

提示：请设置一种可用的背景颜色，这样的话，假如背景图像不可用，页面也可获得良好的视觉效果。

详细说明

background-image 属性会在元素的背景中设置一个图像。

根据 background-repeat 属性的值，图像可以无限平铺、沿着某个轴（x 轴或 y 轴）平铺，或者根本不平铺。

初始背景图像（原图像）根据 background-position 属性的值放置。

默认值：	**none**
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.backgroundImage="url(stars.gif)"
---	---

可能的值

值	描述
url('*URL*')	指向图像的路径。
none	默认值。不显示背景图像。
inherit	规定应该从父元素继承 background-image 属性的设置。

TIY 实例

将图像设置为背景

本例演示如何将图像设置为背景。

```
<html>
<head>

<style type="text/css">
body {background-image:url(/i/eg_bg_04.gif);}
</style>

</head>

<body></body>

</html>
```

相关页面

CSS 教程：CSS 背景

HTML DOM 参考手册：backgroundImage 属性

CSS background-position 属性

实例

如何定位背景图像：

```
body
{
background-image:url('bgimage.gif');
background-repeat:no-repeat;
background-attachment:fixed;
background-position:center;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 background-position 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

background-position 属性设置背景图像的起始位置。

这个属性设置背景原图像（由 background-image 定义）的位置，背景图像如果要重复，将从这一点开始。

提示：您需要把 background-attachment 属性设置为 "fixed"，才能保证该属性在 Firefox 和 Opera 中正常工作。

默认值：	**0% 0%**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.backgroundPosition="center"

可能的值

值	描述

- top left
- top center
- top right
- center left
- center center
- center right
- bottom left
- bottom center
- bottom right

|

如果您仅规定了一个关键词，那么第二个值将是"center"。

默认值：0% 0%。

| | x% y% |

第一个值是水平位置，第二个值是垂直位置。

左上角是 0% 0%。右下角是 100% 100%。

如果您仅规定了一个值，另一个值将是 50%。

| | xpos ypos |

第一个值是水平位置，第二个值是垂直位置。

左上角是 0 0。单位是像素 (0px 0px) 或任何其他的 CSS 单位。

如果您仅规定了一个值，另一个值将是50%。

您可以混合使用 % 和 position 值。

|

TIY 实例

如何放置背景图像

本例演示如何在页面上放置背景图像。

```
<html>
<head>
<style type="text/css">
body
{
  background-image:url('/i/eg_bg_03.gif');
  background-repeat:no-repeat;
  background-attachment:fixed;
  background-position:center;
}
</style>
</head>

<body>
<body>
<p><b>提示：</b>您需要把 background-attachment 属性设置为 "fixed"，才能保
</body>
</body>
</html>
```

如何使用%来定位背景图像

本例演示如何使用百分比来在页面上定位背景图像。

```
<html>
<head>
<style type="text/css">
body
{
background-image: url('/i/eg_bg_03.gif');
background-repeat: no-repeat;
background-attachment:fixed;
background-position: 30% 20%;
}
</style>
</head>

<body>
<p><b>注释：</b>为了在 Mozilla 中实现此效果, background-attachment 属性必
</body>
</html>
```

[如何使用像素来定位背景图像]

本例演示如何使用像素来在页面上定位背景图像。

```
<html>
<head>
<style type="text/css">
body
{
background-image: url('/i/eg_bg_03.gif');
background-repeat: no-repeat;
background-attachment:fixed;
background-position: 50px 100px;
}
</style>
</head>

<body>
<p><b>注释：</b>为了在 Mozilla 中实现此效果, background-attachment 属性必
</body>
</html>
```

相关页面

CSS 教程：CSS 背景

CSS 参考手册：background-image 属性

HTML DOM 参考手册：backgroundPosition 属性

CSS background-repeat 属性

实例

```
body
  {
  background-image: url(stars.gif);
  background-repeat: repeat-y;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 background-repeat 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

background-repeat 属性设置是否及如何重复背景图像。

默认地，背景图像在水平和垂直方向上重复。

详细说明

background-repeat 属性定义了图像的平铺模式。

从原图像开始重复，原图像由 background-image 定义，并根据 background-position 的值放置。

默认值：	**repeat**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.backgroundRepeat="repeat-y"

提示和注释

提示：背景图像的位置是根据 background-position 属性设置的。如果未规定 background-position 属性，图像会被放置在元素的左上角。

可能的值

值	描述
repeat	默认。背景图像将在垂直方向和水平方向重复。
repeat-x	背景图像将在水平方向重复。
repeat-y	背景图像将在垂直方向重复。
no-repeat	背景图像将仅显示一次。
inherit	规定应该从父元素继承 background-repeat 属性的设置。

TIY 实例

如何重复背景图像

本例演示如何重复背景图像。

```
<html>
<head>

<style type="text/css">
body
{
background-image:
url(/i/eg_bg_03.gif);
background-repeat: repeat
}
</style>

</head>

<body>
</body>
</html>
```

如何在垂直方向重复背景图像

本例演示如何垂直地重复背景图像。

```
<html>
<head>

<style type="text/css">
body
{
background-image:
url(/i/eg_bg_03.gif);
background-repeat: repeat-y
}
</style>

</head>

<body>
</body>
</html>
```

如何在水平方向重复背景图像

本例演示如何水平地重复背景图像。

```
<html>
<head>

<style type="text/css">
body
{
background-image:
url(/i/eg_bg_03.gif);
background-repeat: repeat-x
}
</style>

</head>

<body>
</body>
</html>
```

如何仅显示一次背景图像

本例演示如何仅显示一次背景图像。

```
<html>
<head>

<style type="text/css">
body
{
background-image: url('/i/eg_bg_03.gif');
background-repeat: no-repeat
}
</style>

</head>

<body>
</body>
</html>
```

相关页面

CSS 教程：CSS 背景

HTML DOM 参考手册：backgroundRepeat 属性

CSS3 background-clip 属性

实例

规定背景的绘制区域：

```
div
{
background-color:yellow;
background-clip:content-box;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox、Opera、Chrome 以及 Safari 支持 background-clip 属性。

注释：Internet Explorer 8 以及更早的版本不支持 background-clip 属性。

定义和用法

background-clip 属性规定背景的绘制区域。

默认值：	border-box
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.backgroundClip="content-box"
---	---

语法

```
background-clip: border-box|padding-box|content-box;
```

值	描述	测试
border-box	背景被裁剪到边框盒。	测试
padding-box	背景被裁剪到内边距框。	测试
content-box	背景被裁剪到内容框。	测试

相关页面

CSS3 教程：CSS3 背景

CSS3 background-origin 属性

实例

相对于内容框来定位背景图像：

```
div
{
background-image:url('smiley.gif');
background-repeat:no-repeat;
background-position:left;
background-origin:content-box;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Opera、Chrome 以及 Safari 5+ 支持 background-origin 属性。

定义和用法

background-origin 属性规定 background-position 属性相对于什么位置来定位。

注释：如果背景图像的 background-attachment 属性为 "fixed"，则该属性没有效果。

默认值：	padding-box
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.backgroundOrigin="content-box"

语法

```
background-origin: padding-box|border-box|content-box;
```

值	描述	测试
padding-box	背景图像相对于内边距框来定位。	测试
border-box	背景图像相对于边框盒来定位。	测试
content-box	背景图像相对于内容框来定位。	测试

相关页面

CSS3 教程：CSS3 背景

CSS3 background-size 属性

实例

规定背景图像的尺寸：

```
div
{
background:url(img_flwr.gif);
background-size:80px 60px;
background-repeat:no-repeat;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Opera、Chrome 以及 Safari 5+ 支持 background-size 属性。

定义和用法

background-size 属性规定背景图像的尺寸。

默认值：	auto
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.backgroundSize="60px 80px"
---	---

语法

```
background-size: _length_|_percentage_|cover|contain;
```

值	描述	测试
length		

设置背景图像的高度和宽度。

第一个值设置宽度，第二个值设置高度。

如果只设置一个值，则第二个值会被设置为 "auto"。

| 测试 | | *percentage* |

以父元素的百分比来设置背景图像的宽度和高度。

第一个值设置宽度，第二个值设置高度。

如果只设置一个值，则第二个值会被设置为 "auto"。

| 测试 | | cover |

把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。

背景图像的某些部分也许无法显示在背景定位区域中。

| 测试 | | contain | 把图像图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域。 | 测试 |

亲自试一试 - 实例

拉伸背景图像

拉伸背景图像来完全覆盖内容区域。

```
<!DOCTYPE html>
<html>
<head>
<style>
div
{
background:url(/i/bg_flower.gif);
background-size:35% 100%;
-moz-background-size:35% 100%; /* 老版本的 Firefox */
background-repeat:no-repeat;
}
</style>
</head>
<body>

<div>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
<p>这是一个段落。</p>
</div>

</body>
</html>
```

拉伸背景图像，对背景图像水平复制四次

对背景图像进行拉伸，以使背景图像恰好水平复制四次。

```
<!DOCTYPE html>
<html>
<head>
<style>
div
{
background:url(/i/bg_flower.gif);
background-size:25%;
border:2px solid #92b901;
}
</style>
</head>
<body>

<div>
<p>这是一个段落。这是一个段落。这是一个段落。这是一个段落。这是一个段落。</p>
<p>这是一个段落。这是一个段落。这是一个段落。这是一个段落。这是一个段落。</p>
<p>这是一个段落。这是一个段落。这是一个段落。这是一个段落。这是一个段落。</p>
<p>这是一个段落。这是一个段落。这是一个段落。这是一个段落。这是一个段落。</p>
<p>这是一个段落。这是一个段落。这是一个段落。这是一个段落。这是一个段落。</p>
<p>这是一个段落。这是一个段落。这是一个段落。这是一个段落。这是一个段落。</p>
</div>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 背景

CSS 边框属性（Border 和 Outline）

属性	描述	CSS
border	在一个声明中设置所有的边框属性。	1
border-bottom	在一个声明中设置所有的下边框属性。	1
border-bottom-color	设置下边框的颜色。	2
border-bottom-style	设置下边框的样式。	2
border-bottom-width	设置下边框的宽度。	1
border-color	设置四条边框的颜色。	1
border-left	在一个声明中设置所有的左边框属性。	1
border-left-color	设置左边框的颜色。	2
border-left-style	设置左边框的样式。	2
border-left-width	设置左边框的宽度。	1
border-right	在一个声明中设置所有的右边框属性。	1
border-right-color	设置右边框的颜色。	2
border-right-style	设置右边框的样式。	2
border-right-width	设置右边框的宽度。	1
border-style	设置四条边框的样式。	1
border-top	在一个声明中设置所有的上边框属性。	1
border-top-color	设置上边框的颜色。	2
border-top-style	设置上边框的样式。	2
border-top-width	设置上边框的宽度。	1
border-width	设置四条边框的宽度。	1
outline	在一个声明中设置所有的轮廓属性。	2
outline-color	设置轮廓的颜色。	2
outline-style	设置轮廓的样式。	2
outline-width	设置轮廓的宽度。	2

border-bottom-left-radius	定义边框左下角的形状。	3
border-bottom-right-radius	定义边框右下角的形状。	3
border-image	简写属性，设置所有 border-image-* 属性。	3
border-image-outset	规定边框图像区域超出边框的量。	3
border-image-repeat	图像边框是否应平铺(repeated)、铺满(rounded)或拉伸(stretched)。	3
border-image-slice	规定图像边框的向内偏移。	3
border-image-source	规定用作边框的图片。	3
border-image-width	规定图片边框的宽度。	3
border-radius	简写属性，设置所有四个 border-*-radius 属性。	3
border-top-left-radius	定义边框左上角的形状。	3
border-top-right-radius	定义边框右下角的形状。	3
box-decoration-break	3	
box-shadow	向方框添加一个或多个阴影。	3

CSS border 属性

实例

设置 4 个边框的样式：

```
p
  {
  border:5px solid red;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border 简写属性在一个声明设置所有的边框属性。

可以按顺序设置如下属性：

- border-width
- border-style
- border-color

如果不设置其中的某个值，也不会出问题，比如 border:solid #ff0000; 也是允许的。

默认值：	*not specified*
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.border="3px solid blue"
---	---

可能的值

值	描述
border-width	规定边框的宽度。参阅：border-width 中可能的值。
border-style	规定边框的样式。参阅：border-style 中可能的值。
border-color	规定边框的颜色。参阅：border-color 中可能的值。
inherit	规定应该从父元素继承 border 属性的设置。

TIY 实例

所有边框属性在一个声明之中

本例演示用简写属性来将所有四个边框属性设置于同一声明中。

```
<html>
<head>
<style type="text/css">
p
{
border: medium double rgb(250,0,255)
}
</style>
</head>

<body>
<p>Some text</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：border 属性

CSS border-bottom 属性

实例

设置下边框的样式：

```
p
  {
  border-style:solid;
  border-bottom:thick dotted #ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-bottom 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-bottom 简写属性把下边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

- border-bottom-width
- border-bottom-style
- border-bottom-color

如果不设置其中的某个值，也不会出问题，比如 border-bottom:solid #ff0000; 也是允许的。

默认值：	***not specified***
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderBottom="3px solid blue"
---	---

可能的值

值	描述
border-bottom-width	规定下边框的宽度。参阅：border-bottom-width 中可能的值。
border-bottom-style	规定下边框的样式。参阅：border-bottom-style 中可能的值。
border-bottom-color	规定下边框的颜色。参阅：border-bottom-color 中可能的值。
inherit	规定应该从父元素继承 border-bottom 属性的设置。

TIY 实例

所有下边框属性在一个声明中

本例演示用简写属性来将所有下边框属性设置在同一声明中。

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
border-bottom:thick dotted #ff0000;
}
</style>
</head>

<body>
<p>This is some text in a paragraph.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderBottom 属性

CSS border-bottom-color 属性

实例

设置下边框的颜色：

```
p
  {
  border-style:solid;
  border-bottom-color:#ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-bottom-color 属性。

注释：Internet Explorer 6（以及更早的版本）不支持属性值 "transparent"。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-bottom-color 设置元素的下边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值：	***not specified***
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderBottomColor="blue"
---	---

可能的值

值	描述
color_name	规定颜色值为颜色名称的边框颜色（比如 red）。
hex_number	规定颜色值为十六进制值的边框颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。
transparent	默认值。边框颜色为透明。
inherit	规定应该从父元素继承边框颜色。

TIY 实例

设置下边框的颜色

本例演示如何设置下边框的颜色。

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
border-bottom-color:#ff0000;
}
</style>
</head>

<body>
<p>This is some text in a paragraph.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-bottom 属性

HTML DOM 参考手册：borderBottomColor 属性

CSS border-bottom-style 属性

实例

设置下边框的样式：

```
p
  {
  border-style:solid;
  border-bottom-style:dotted;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-bottom-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit" 或 "hidden"。

定义和用法

border-bottom-style 设置元素下边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值：	*not specified*
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderBottomStyle="dotted"
---	---

可能的值

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

TIY 实例

设置下边框的样式

本例演示如何设置下边框的样式。

```
<html>
<head>
<style type="text/css">
p {border-style:solid}
p.none {border-bottom-style:none}
p.dotted {border-bottom-style:dotted}
p.dashed {border-bottom-style:dashed}
p.solid {border-bottom-style:solid}
p.double {border-bottom-style:double}
p.groove {border-bottom-style:groove}
p.ridge {border-bottom-style:ridge}
p.inset {border-bottom-style:inset}
p.outset {border-bottom-style:outset}
</style>
</head>

<body>
<p class="none">No bottom border.</p>
<p class="dotted">A dotted bottom border.</p>
<p class="dashed">A dashed bottom border.</p>
<p class="solid">A solid bottom border.</p>
<p class="double">A double bottom border.</p>
<p class="groove">A groove bottom border.</p>
<p class="ridge">A ridge bottom border.</p>
<p class="inset">An inset bottom border.</p>
<p class="outset">An outset bottom border.</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-bottom 属性

HTML DOM 参考手册：borderBottomStyle 属性

CSS border-bottom-width 属性

实例

设置下边框的宽度：

```
p
  {
  border-style:solid;
  border-bottom-width:15px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-bottom-width 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-bottom-width 属性设置元素的下边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-bottom-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值：	medium
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderBottomWidth="thick"
---	---

可能的值

值	描述
thin	定义细的下边框。
medium	默认值。定义中等的下边框。
thick	定义粗的下边框。
length	允许您自定义下边框的宽度。
inherit	规定应该从父元素继承边框宽度。

TIY 实例

设置下边框的宽度

本例演示如何设置下边框的宽度。

```
<html>
<head>
<style type="text/css">
p.one
{
border-style: solid;
border-bottom-width: 15px
}
p.two
{
border-style: solid;
border-bottom-width: thin
}
</style>
</head>
<body>

<p class="one"><b>注释：</b>"border-bottom-width" 属性如果单独使用的话是
<p class="two">Some text. Some more text.</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-bottom 属性

HTML DOM 参考手册：borderBottomWidth 属性

CSS border-color 属性

实例

设置 4 个边框的颜色：

```
p
  {
  border-style:solid;
  border-color:#ff0000 #0000ff;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-color 属性。

注释：Internet Explorer 6（以及更早的版本）不支持属性值 "transparent"。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-color 属性设置四条边框的颜色。此属性可设置 1 到 4 种颜色。

border-color 属性是一个简写属性，可设置一个元素的所有边框中可见部分的颜色，或者为 4 个边分别设置不同的颜色。请看下面的例子：

例子 1

```
border-color:red green blue pink;
```

- 上边框是红色
- 右边框是绿色
- 下边框是蓝色
- 左边框是粉色

例子 2

```
border-color:red green blue;
```

- 上边框是红色
- 右边框和左边框是绿色
- 下边框是蓝色

例子 3

```
border-color:dotted red green;
```

- 上边框和下边框是红色
- 右边框和左边框是绿色

例子 4

```
border-color:red;
```

- 所有 4 个边框都是红色

要记住，边框的样式不能为 none 或 hidden，否则边框不会出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderColor="#FF0000 blue"

可能的值

值	描述
color_name	规定颜色值为颜色名称的边框颜色（比如 red）。
hex_number	规定颜色值为十六进制值的边框颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。
transparent	默认值。边框颜色为透明。
inherit	规定应该从父元素继承边框颜色。

TIY 实例

设置四个边框的颜色

本例演示如何设置四个边框的颜色。可以设置一到四种颜色。

```
<html>
<head>

<style type="text/css">
p.one
{
border-style: solid;
border-color: #0000ff
}
p.two
{
border-style: solid;
border-color: #ff0000 #0000ff
}
p.three
{
border-style: solid;
border-color: #ff0000 #00ff00 #0000ff
}
p.four
{
border-style: solid;
border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255)
}
</style>

</head>

<body>

<p class="one">One-colored border!</p>

<p class="two">Two-colored border!</p>

<p class="three">Three-colored border!</p>

<p class="four">Four-colored border!</p>

<p><b>注释：</b>"border-width" 属性如果单独使用的话是不会起作用的。请首先使

</body>
</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderColor 属性

CSS border-left 属性

实例

设置左边框的样式：

```
p
  {
  border-style:solid;
  border-left:thick double #ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-left 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-left 简写属性把左边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

- border-left-width
- border-left-style
- border-left-color

如果不设置其中的某个值，也不会出问题，比如 border-left:solid #ff0000; 也是允许的。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderLeft="3px solid blue"

可能的值

值	描述
border-left-width	规定左边框的宽度。参阅：border-left-width 中可能的值。
border-left-style	规定左边框的样式。参阅：border-left-style 中可能的值。
border-left-color	规定左边框的颜色。参阅：border-left-color 中可能的值。
inherit	规定应该从父元素继承 border-left 属性的设置。

TIY 实例

所有左边框属性在一个声明之中

所有左边框属性在一个声明之中

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
border-left:thick double #ff0000;
}
</style>
</head>

<body>
<p>This is some text in a paragraph.</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderLeft 属性

CSS border-left-color 属性

实例

设置左边框的颜色：

```
p
  {
  border-style:solid;
  border-left-color:#ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-left-color 属性。

注释：Internet Explorer 6（以及更早的版本）不支持属性值 "transparent"。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-left-color 设置元素的左边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值：	***not specified***
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderLeftColor="blue"
---	---

可能的值

值	描述
color_name	规定颜色值为颜色名称的边框颜色（比如 red）。
hex_number	规定颜色值为十六进制值的边框颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。
transparent	默认值。边框颜色为透明。
inherit	规定应该从父元素继承边框颜色。

TIY 实例

设置左边框的颜色

本例演示如何设置左边框的颜色。

```
<html>
<head>
<style type="text/css">
p
{
border-style: solid;
border-left-color: #ff0000
}
</style>
</head>

<body>
<p>Some text.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-left 属性

HTML DOM 参考手册：borderLeftColor 属性

CSS border-left-style 属性

实例

设置左边框的样式：

```
p
  {
  border-style:solid;
  border-left-style:dotted;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-left-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit" 或 "hidden"。

定义和用法

border-left-style 设置元素左边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderLeftStyle="dotted"

可能的值

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

TIY 实例

设置左边框的样式

本例演示如何设置左边框的样式。

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
}
p.none {border-left-style:none}
p.dotted {border-left-style:dotted}
p.dashed {border-left-style:dashed}
p.solid {border-left-style:solid}
p.double {border-left-style:double}
p.groove {border-left-style:groove}
p.ridge {border-left-style:ridge}
p.inset {border-left-style:inset}
p.outset {border-left-style:outset}
</style>
</head>

<body>
<p class="none">No left border.</p>
<p class="dotted">A dotted left border.</p>
<p class="dashed">A dashed left border.</p>
<p class="solid">A solid left border.</p>
<p class="double">A double left border.</p>
<p class="groove">A groove left border.</p>
<p class="ridge">A ridge left border.</p>
<p class="inset">An inset left border.</p>
<p class="outset">An outset left border.</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-left 属性

HTML DOM 参考手册：borderLeftStyle 属性

CSS border-left-width 属性

实例

设置左边框的宽度：

```
p
  {
  border-style:solid;
  border-left-width:15px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-left-width 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-left-width 属性设置元素的左边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-left-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值：	medium
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderLeftWidth="thick"

可能的值

值	描述
thin	定义细的左边框。
medium	默认值。定义中等的左边框。
thick	定义粗的左边框。
length	允许您自定义左边框的宽度。
inherit	规定应该从父元素继承边框宽度。

TIY 实例

设置左边框的宽度

本例演示如何设置左边框的宽度。

```
<html>
<head>
<style type="text/css">
p.one
{
border-style: solid;
border-left-width: 15px
}
p.two
{
border-style: solid;
border-left-width: thin
}
</style>
</head>
<body>

<p class="one"><b>注释：</b>"border-left-width" 属性如果单独使用的话是不
<p class="two">Some text. Some more text.</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-left 属性

HTML DOM 参考手册：borderLeftWidth 属性

CSS border-right 属性

实例

设置右边框的样式：

```
p
  {
  border-style:solid;
  border-right:thick double #ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-right 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-right 简写属性把右边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

- border-right-width
- border-right-style
- border-right-color

如果不设置其中的某个值，也不会出问题，比如 border-right:solid #ff0000; 也是允许的。

默认值：	***not specified***
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderRight="3px solid blue"
---	---

可能的值

值	描述
border-right-width	规定右边框的宽度。参阅：border-right-width 中可能的值。
border-right-style	规定右边框的样式。参阅：border-right-style 中可能的值。
border-right-color	规定右边框的颜色。参阅：border-right-color 中可能的值。
inherit	规定应该从父元素继承 border-right 属性的设置。

TIY 实例

所有右边框属性在一个声明之中

本例演示一个简写属性，用于把所有右边框属性设置在一条声明中。

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
border-right:thick double #ff0000;
}
</style>
</head>

<body>
<p>This is some text in a paragraph.</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderRight 属性

CSS border-right-color 属性

实例

设置右边框的颜色：

```
p
  {
  border-style:solid;
  border-right-color:#ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-right-color 属性。

注释：Internet Explorer 6（以及更早的版本）不支持属性值 "transparent"。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-right-color 设置元素的右边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderRightColor="blue"

可能的值

值	描述
color_name	规定颜色值为颜色名称的边框颜色（比如 red）。
hex_number	规定颜色值为十六进制值的边框颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。
transparent	默认值。边框颜色为透明。
inherit	规定应该从父元素继承边框颜色。

TIY 实例

设置右边框的颜色

本例演示如何设置右边框的颜色。

```
<html>
<head>
<style type="text/css">
p
{
border-style: solid;
border-right-color: #ff0000
}
</style>
</head>

<body>
<p>Some text.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-right 属性

HTML DOM 参考手册：borderRightColor 属性

CSS border-right-style 属性

实例

设置右边框的样式：

```
p
  {
  border-style:solid;
  border-right-style:dotted;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-right-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit" 或 "hidden"。

定义和用法

border-right-style 设置元素右边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderRightStyle="dotted"

可能的值

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

TIY 实例

设置右边框的样式

本例演示如何设置右边框的样式。

```
<!DOCTYPE html>

<html>
<head>
<style type="text/css">
p.dotted {border-right-style: dotted}
p.dashed {border-right-style: dashed}
p.solid {border-right-style: solid}
p.double {border-right-style: double}
p.groove {border-right-style: groove}
p.ridge {border-right-style: ridge}
p.inset {border-right-style: inset}
p.outset {border-right-style: outset}
</style>
</head>

<body>
<p class="dotted">A dotted border</p>

<p class="dashed">A dashed border</p>

<p class="solid">A solid border</p>

<p class="double">A double border</p>

<p class="groove">A groove border</p>

<p class="ridge">A ridge border</p>

<p class="inset">An inset border</p>

<p class="outset">An outset border</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-right 属性

HTML DOM 参考手册：borderRightStyle 属性

CSS border-right-width 属性

实例

设置右边框的宽度：

```
p
  {
  border-style:solid;
  border-right-width:15px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-right-width 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-right-width 属性设置元素的右边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-right-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值：	medium
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderRightWidth="thick"

可能的值

值	描述
thin	定义细的右边框。
medium	默认值。定义中等的右边框。
thick	定义粗的右边框。
length	允许您自定义右边框的宽度。
inherit	规定应该从父元素继承边框宽度。

TIY 实例

设置右边框的宽度

本例演示如何设置右边框的宽度。

```
<html>
<head>
<style type="text/css">
p.one
{
border-style: solid;
border-right-width: 15px
}
p.two
{
border-style: solid;
border-right-width: thin
}
</style>
</head>
<body>

<p class="one"><b>注释：</b>"border-right-width" 属性如果单独使用的话是
<p class="two">Some text. Some more text.</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-right 属性

HTML DOM 参考手册：borderRightWidth 属性

CSS border-style 属性

实例

设置 4 个边框的样式：

```
p
  {
  border-style:solid;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit" 或 "hidden"。

定义和用法

border-style 属性用于设置元素所有边框的样式，或者单独地为各边设置边框样式。

只有当这个值不是 none 时边框才可能出现。

例子 1

```
border-style:dotted solid double dashed;
```

- 上边框是点状
- 右边框是实线
- 下边框是双线
- 左边框是虚线

例子 2

```
border-style:dotted solid double;
```

- 上边框是点状
- 右边框和左边框是实线
- 下边框是双线

例子 3

```
border-style:dotted solid;
```

- 上边框和下边框是点状
- 右边框和左边框是实线

例子 4

```
border-style:dotted;
```

- 所有 4 个边框都是点状

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderStyle="dotted double"

可能的值

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

描述

最不可预测的边框样式是 double。它定义为两条线的宽度再加上这两条线之间的空间等于 border-width 值。不过，CSS 规范并没有说其中一条线是否比另一条粗或者两条线是否应该是一样的粗，也没有指出线之间的空间是否应当比线粗。所有这些都有用户代理决定，创作人员对这个决定没有任何影响。

TIY 实例

设置四边框样式

本例演示如何设置四边框样式。

```
<html>
<head>
<style type="text/css">
p.dotted {border-style: dotted}
p.dashed {border-style: dashed}
p.solid {border-style: solid}
p.double {border-style: double}
p.groove {border-style: groove}
p.ridge {border-style: ridge}
p.inset {border-style: inset}
p.outset {border-style: outset}
</style>
</head>

<body>
<p class="dotted">A dotted border</p>

<p class="dashed">A dashed border</p>

<p class="solid">A solid border</p>

<p class="double">A double border</p>

<p class="groove">A groove border</p>

<p class="ridge">A ridge border</p>

<p class="inset">An inset border</p>

<p class="outset">An outset border</p>
</body>

</html>
```

设置每一边的不同边框

本例演示如何在元素的各边设置不同的边框。

```
<html>
<head>
<style type="text/css">
p.soliddouble {border-style: solid double}
p.doublesolid {border-style: double solid}
p.groovedouble {border-style: groove double}
p.three {border-style: solid double groove}
</style>
</head>

<body>
<p class="soliddouble">Some text</p>

<p class="doublesolid">Some text</p>

<p class="groovedouble">Some text</p>

<p class="three">Some text</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderStyle 属性

CSS border-top 属性

实例

设置上边框的样式：

```
p
  {
  border-style:solid;
  border-top:thick double #ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-top 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-top 简写属性把上边框的所有属性设置到一个声明中。

可以按顺序设置如下属性：

- border-top-width
- border-top-style
- border-top-color

如果不设置其中的某个值，也不会出问题，比如 border-top:solid #ff0000; 也是允许的。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderTop="3px solid blue"

可能的值

值	描述
border-top-width	规定上边框的宽度。参阅：border-top-width 中可能的值。
border-top-style	规定上边框的样式。参阅：border-top-style 中可能的值。
border-top-color	规定上边框的颜色。参阅：border-top-color 中可能的值。
inherit	规定应该从父元素继承 border-top 属性的设置。

TIY 实例

所有上边框属性在一个声明之中

本例演示用简写属性来将所有上边框属性设置于同一声明之中。

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
border-top:thick double #ff0000;
}
</style>
</head>

<body>
<p>This is some text in a paragraph.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderTop 属性

CSS border-top-color 属性

实例

设置上边框的颜色：

```
p
  {
  border-style:solid;
  border-top-color:#ff0000;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-top-color 属性。

注释：Internet Explorer 6（以及更早的版本）不支持属性值 "transparent"。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-top-color 设置元素的上边框的颜色。

只能定义纯色，而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现。

注释：请始终把 border-style 属性声明到 border-color 属性之前。元素必须在您改变其颜色之前获得边框。

默认值：	***not specified***
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderTopColor="blue"

可能的值

值	描述
color_name	规定颜色值为颜色名称的边框颜色（比如 red）。
hex_number	规定颜色值为十六进制值的边框颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的边框颜色（比如 rgb(255,0,0)）。
transparent	默认值。边框颜色为透明。
inherit	规定应该从父元素继承边框颜色。

TIY 实例

设置上边框的颜色

本例演示如何设置上边框的颜色。

```
<html>
<head>
<style type="text/css">
p
{
border-style: solid;
border-top-color: #ff0000
}
</style>
</head>

<body>
<p>Some text.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-top 属性

HTML DOM 参考手册：borderTopColor 属性

CSS border-top-style 属性

实例

设置上边框的样式：

```
p
  {
  border-style:solid;
  border-top-style:dotted;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-top-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit" 或 "hidden"。

定义和用法

border-top-style 设置元素上边框的样式。

只有当这个值不是 none 时边框才可能出现。

在 CSS1 中，HTML 用户代理只需支持 solid 和 none。

默认值：	***not specified***
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.borderTopStyle="dotted"
---	---

可能的值

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

TIY 实例

设置上边框的样式

本例演示如何设置上边框的样式。

```
<html>
<head>
<style type="text/css">
p
{
border-style:solid;
}
p.none {border-top-style:none}
p.dotted {border-top-style:dotted}
p.dashed {border-top-style:dashed}
p.solid {border-top-style:solid}
p.double {border-top-style:double}
p.groove {border-top-style:groove}
p.ridge {border-top-style:ridge}
p.inset {border-top-style:inset}
p.outset {border-top-style:outset}
</style>
</head>

<body>
<p class="none">No top border.</p>
<p class="dotted">A dotted top border.</p>
<p class="dashed">A dashed top border.</p>
<p class="solid">A solid top border.</p>
<p class="double">A double top border.</p>
<p class="groove">A groove top border.</p>
<p class="ridge">A ridge top border.</p>
<p class="inset">An inset top border.</p>
<p class="outset">An outset top border.</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-top 属性

HTML DOM 参考手册：borderTopStyle 属性

CSS border-top-width 属性

实例

设置上边框的宽度：

```
p
  {
  border-style:solid;
  border-top-width:15px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-top-width 属性。

注释：IE7 以及更早版本的浏览器不支持值 "inherit"。IE8 需要 !DOCTYPE。IE9 支持 "inherit"。

定义和用法

border-top-width 属性设置元素的上边框的宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

注释：请始终在 border-top-width 属性之前声明 border-style 属性。元素只有在获得边框之后，才能改变其边框的宽度。

默认值：	medium
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderTopWidth="thick"

可能的值

值	描述
thin	定义细的上边框。
medium	默认值。定义中等的上边框。
thick	定义粗的上边框。
length	允许您自定义上边框的宽度。
inherit	规定应该从父元素继承边框宽度。

TIY 实例

设置上边框的宽度

本例演示如何设置上边框的宽度。

```
<html>
<head>
<style type="text/css">
p.one
{
border-style: solid;
border-top-width: 15px
}
p.two
{
border-style: solid;
border-top-width: thin
}
</style>
</head>

<body>
<p class="one"><b>注释：</b>"border-top-width" 属性如果单独使用的话是不
<p class="two">Some text. Some more text.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

CSS 参考手册：border-top 属性

HTML DOM 参考手册：borderTopWidth 属性

CSS border-width 属性

实例

设置四个边框的宽度：

```
p
  {
  border-style:solid;
  border-width:15px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 border-width 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

border-width 简写属性为元素的所有边框设置宽度，或者单独地为各边边框设置宽度。

只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。

例子 1

```
border-width:thin medium thick 10px;
```

- 上边框是细边框
- 右边框是中等边框
- 下边框是粗边框
- 左边框是 10px 宽的边框

例子 2

```
border-width:thin medium thick;
```

- 上边框是 10px
- 右边框和左边框是中等边框
- 下边框是粗边框

例子 3

```
border-width:thin medium;
```

- 上边框和下边框是细边框
- 右边框和左边框是中等边框

例子 4

```
border-width:thin;
```

- 所有 4 个边框都是细边框

默认值：	**medium**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.borderWidth="thin thick"

可能的值

值	描述
thin	定义细的边框。
medium	默认。定义中等的边框。
thick	定义粗的边框。
length	允许您自定义边框的宽度。
inherit	规定应该从父元素继承边框宽度。

TIY 实例

所有边框宽度属性在一个声明之中

本例演示用简写属性来将所有边框宽度属性设置于同一声明中。

```
<html>
<head>
<style type="text/css">
p.one
{
border-style: solid;
border-width: 5px
}
p.two
{
border-style: solid;
border-width: thick
}
p.three
{
border-style: solid;
border-width: 5px 10px
}
p.four
{
border-style: solid;
border-width: 5px 10px 1px
}
p.five
{
border-style: solid;
border-width: 5px 10px 1px medium
}
</style>
</head>

<body>
<p class="one">Some text</p>
<p class="two">Some text</p>
<p class="three">Some text</p>
<p class="four">Some text</p>
<p class="five">Some text</p>

<p><b>注释：</b>"border-width" 属性如果单独使用的话是不会起作用的。请首先使
</body>

</html>
```

相关页面

CSS 教程：CSS 边框

HTML DOM 参考手册：borderWidth 属性

CSS outline 属性

实例

设置 4 个边框的样式：

```
p
  {
  outline:#00FF00 dotted thick;
  }
```

浏览器支持

所有浏览器都支持 outline 属性。

注释：如果规定了 !DOCTYPE，则 IE8 支持 outline 属性。

定义和用法

outline （轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

注释：轮廓线不会占据空间，也不一定是矩形。

outline 简写属性在一个声明中设置所有的轮廓属性。

可以按顺序设置如下属性：

- outline-color
- outline-style
- outline-width

如果不设置其中的某个值，也不会出问题，比如 outline:solid #ff0000; 也是允许的。

默认值：	invert none medium
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.outline="#0000FF dotted thin"
---	---

可能的值

值	描述
outline-color	规定边框的颜色。参阅：outline-color 中可能的值。
outline-style	规定边框的样式。参阅：outline-style 中可能的值。
outline-width	规定边框的宽度。参阅：outline-width 中可能的值。
inherit	规定应该从父元素继承 outline 属性的设置。

TIY 实例

在元素周围画线

本例演示使用 outline 属性在元素周围画一条线。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
<html>
<head>
<style type="text/css">
p
{
border:red solid thin;
outline:#00ff00 dotted thick;
}
</style>
</head>

<body>
<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高
</body>
</html>
```

相关页面

CSS 教程：CSS 轮廓

HTML DOM 参考手册：outline 属性

CSS outline-color 属性

实例

设置点状轮廓的颜色：

```
p
  {
  outline-style:dotted;
  outline-color:#00ff00;
  }
```

浏览器支持

所有浏览器都支持 outline-color 属性。

注释：如果规定了 !DOCTYPE，则 IE8 支持 outline 属性。

定义和用法

outline （轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。outline 属性可设置元素周围的轮廓线。

注释：请始终在 outline-color 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

注释：轮廓线不会占据空间，也不一定是矩形。

outline-color 属性设置一个元素整个轮廓中可见部分的颜色。要记住，轮廓的样式不能是 none，否则轮廓不会出现。

默认值：	invert
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.outlineColor="#0000FF"
---	---

可能的值

值	描述
color_name	规定颜色值为颜色名称的轮廓颜色（比如 red）。
hex_number	规定颜色值为十六进制值的轮廓颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的轮廓颜色（比如 rgb(255,0,0)）。
invert	默认。执行颜色反转（逆向的颜色）。可使轮廓在不同的背景颜色中都是可见。
inherit	规定应该从父元素继承轮廓颜色的设置。

TIY 实例

设置轮廓的颜色

本例演示如何设置轮廓的颜色。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
<html>
<head>
<style type="text/css">
p
{
border:red solid thin;
outline-style:dotted;
outline-color:#00ff00;
}
</style>
</head>

<body>
<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高
</body>
</html>
```

相关页面

CSS 教程：CSS 轮廓

CSS 参考手册：outline 属性

HTML DOM 参考手册：outlineColor 属性

CSS outline-style 属性

实例

设置轮廓的样式：

```
p
  {
  outline-style:dotted;
  }
```

浏览器支持

所有浏览器都支持 outline-style 属性。

注释：如果规定了 !DOCTYPE，则 IE8 支持 outline 属性。

定义和用法

outline-style 属性用于设置元素的整个轮廓的样式。样式不能是 none，否则轮廓不会出现。

outline （轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。outline 属性设置元素周围的轮廓线。

注释：请始终在 outline-color 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

注释：轮廓线不会占据空间，也不一定是矩形。

默认值：	none
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.outlineStyle="dotted"

可能的值

值	描述
none	默认。定义无轮廓。
dotted	定义点状的轮廓。
dashed	定义虚线轮廓。
solid	定义实线轮廓。
double	定义双线轮廓。双线的宽度等同于 outline-width 的值。
groove	定义 3D 凹槽轮廓。此效果取决于 outline-color 值。
ridge	定义 3D 凸槽轮廓。此效果取决于 outline-color 值。
inset	定义 3D 凹边轮廓。此效果取决于 outline-color 值。
outset	定义 3D 凸边轮廓。此效果取决于 outline-color 值。
inherit	规定应该从父元素继承轮廓样式的设置。

TIY 实例

设置轮廓的样式

本例演示如何设置轮廓的样式。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
<html>
<head>
<style type="text/css">
p
{
border: red solid thin;
}
p.dotted {outline-style: dotted}
p.dashed {outline-style: dashed}
p.solid {outline-style: solid}
p.double {outline-style: double}
p.groove {outline-style: groove}
p.ridge {outline-style: ridge}
p.inset {outline-style: inset}
p.outset {outline-style: outset}
</style>
</head>
<body>

<p class="dotted">A dotted outline</p>
<p class="dashed">A dashed outline</p>
<p class="solid">A solid outline</p>
<p class="double">A double outline</p>
<p class="groove">A groove outline</p>
<p class="ridge">A ridge outline</p>
<p class="inset">An inset outline</p>
<p class="outset">An outset outline</p>

<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高
</body>
</html>
```

相关页面

CSS 教程：CSS 轮廓

CSS 参考手册：outline 属性

HTML DOM 参考手册：outlineStyle 属性

CSS outline-width 属性

实例

设置点状轮廓的颜色：

```
p
  {
  outline-style:dotted;
  outline-width:5px;
  }
```

浏览器支持

所有浏览器都支持 outline-width 属性。

注释：如果规定了 !DOCTYPE，则 IE8 支持 outline 属性。

定义和用法

outline-width 属性设置元素整个轮廓的宽度，只有当轮廓样式不是 none 时，这个宽度才会起作用。如果样式为 none，宽度实际上会重置为 0。不允许设置负长度值。

outline（轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。outline 属性设置元素周围的轮廓线。

注释：请始终在 outline-width 属性之前声明 outline-style 属性。元素只有获得轮廓以后才能改变其轮廓的颜色。

注释：轮廓线不会占据空间，也不一定是矩形。

默认值：	medium
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.outlineWidth="thin"

可能的值

值	描述
thin	规定细轮廓。
medium	默认。规定中等的轮廓。
thick	规定粗的轮廓。
length	允许您规定轮廓粗细的值。
inherit	规定应该从父元素继承轮廓宽度的设置。

TIY 实例

设置轮廓的宽度

本例演示如何设置轮廓的宽度。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
<html>
<head>
<style type="text/css">
p.one
{
border:red solid thin;
outline-style:solid;
outline-width:thin;
}
p.two
{
border:red solid thin;
outline-style:dotted;
outline-width:3px;
}
</style>
</head>
<body>

<p class="one">This is some text in a paragraph.</p>
<p class="two">This is some text in a paragraph.</p>

<p><b>注释：</b>只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高
</body>
</html>
```

相关页面

CSS 教程：CSS 轮廓

CSS 参考手册：outline 属性

HTML DOM 参考手册：outlineWidth 属性

CSS3 border-bottom-left-radius 属性

实例

向 div 元素的左下角添加圆角边框：

```
div
{
border:2px solid;
border-bottom-left-radius:2em;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-bottom-left-radius 属性。

定义和用法

border-bottom-left-radius 属性定义左下角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderBottomLeftRadius="5px"

语法

```
border-bottom-left-radius: _length_|_%_ [_length_|_%_];
```

注释：border-bottom-left-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

值	描述	测试
length	定义左下角的形状。	测试
%	以百分比值定义左下角的形状。	测试

相关页面

CSS3 教程：CSS3 边框

CSS3 border-bottom-right-radius 属性

实例

向 div 元素的右下角添加圆角边框：

```
div
{
border:2px solid;
border-bottom-right-radius:2em;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-bottom-right-radius 属性。

定义和用法

border-bottom-right-radius 属性定义右下角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderBottomRightRadius="5px"

语法

```
border-bottom-right-radius: _length_|_%_ [_length_|_%_];
```

注释：border-bottom-right-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

值	描述	测试
length	定义右下角的形状。	测试
%	以百分比值定义右下角的形状。	测试

相关页面

CSS3 教程：CSS3 边框

CSS3 border-image 属性

实例

将图片规定为包围 div 元素的边框：

```
div
{
-webkit-border-image:url(border.png) 30 30 round; /* Safari 5 */
-o-border-image:url(border.png) 30 30 round; /* Opera */
border-image:url(border.png) 30 30 round;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 11, Firefox, Opera 15, Chrome 以及 Safari 6 支持 border-image 属性。

Safari 5 支持替代的 -webkit-border-image 属性。

定义和用法

border-image 属性是一个简写属性，用于设置以下属性：

- border-image-source
- border-image-slice
- border-image-width
- border-image-outset
- border-image-repeat

如果省略值，会设置其默认值。

提示：请使用 border-image-* 属性来构造漂亮的可伸缩按钮！

默认值：	**none 100% 1 0 stretch**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderImage="url(border.png) 30 30 round"

可能的值

值	描述	测试
border-image-source	用在边框的图片的路径。	
border-image-slice	图片边框向内偏移。	
border-image-width	图片边框的宽度。	
border-image-outset	边框图像区域超出边框的量。	
border-image-repeat	图像边框是否应平铺(repeated)、铺满(rounded)或拉伸(stretched)。	测试

亲自试一试 - 实例

Border-image 按钮

本例演示如何通过 border-image 属性来创建按钮。

```
<!DOCTYPE html>
<html>
<head>
<style>
div
{
border:10px solid transparent;
width:40px;
padding:5px 10px;
-moz-border-image: url(/i/border_image_button.png) 0 14 0 14 stretc
-webkit-border-image: url(/i/border_image_button.png) 0 14 0 14 str
-o-border-image: url(/i/border_image_button.png) 0 14 0 14 stretch,
border-image: url(/i/border_image_button.png) 0 14 0 14 stretch;
}
</style>
</head>
<body>

<p><b>注释：</b>Internet Explorer 不支持 border-image 属性。</p>

<div>Search</div>

<p>这是我们使用的图片：</p>
<img src="/i/border_image_button.png">

</body>
</html>
```

相关页面

CSS3 教程：CSS3 边框

CSS3 border-image-outset 属性

实例

设置 border-image-outset 属性：

```
div
{
border-image-source: url(border.png);
border-image-outset: 30 30;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Opera 不支持 border-image-outset 属性。

Internet Explorer 10 以及更早的版本不支持 border-image-outset 属性。

Safari 5 以及更早的版本不支持 border-image-outset 属性。

请参阅 border-image 属性。

定义和用法

border-image-outset 属性规定边框图像超过边框盒的量。

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderImageOutset="30 30"

可能的值

```
border-image-outset: _length_|_number_;
```

注释：border-image-outset 属性规定边框图像超出边框盒的量。在上、右、下、左侧。如果忽略第四个值，则与第二个值相同。如果省略第三个值，则与第一个值相同。如果省略第二个值，则与第一个值相同。不允许任何负值作为 border-image-outset 值。

值	描述
length	
number	代表对应的 border-width 的倍数。

相关页面

CSS3 教程：CSS3 边框

CSS3 border-image-repeat 属性

实例

规定如何重复图像边框：

```
div
{
border-image-source: url(border.png);
border-image-repeat: round;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Opera 不支持 border-image-repeat 属性。

Internet Explorer 10 以及更早的版本不支持 border-image-repeat 属性。

Safari 5 以及更早的版本不支持 border-image-repeat 属性。

请参阅 border-image 属性。

定义和用法

border-image-repeat 属性规定图像边框是否应该被重复（repeated）、拉伸（stretched）或铺满（rounded）。

默认值：	stretch
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderImageRepeat="round"

可能的值

```
border-image-repeat: stretch|repeat|round;
```

注释：该属性规定如何延展和铺排边框图像的边缘和中间部分。因此，您可以规定两个值。如果省略第二个值，则采取与第一个值相同的值。

值	描述
stretch	拉伸图像来填充区域
repeat	平铺（重复）图像来填充区域。
round	类似 repeat 值。如果无法完整平铺所有图像，则对图像进行缩放以适应区域。

相关页面

CSS3 教程：CSS3 边框

CSS3 border-image-slice 属性

实例

规定图像边框的向内偏移：

```
div
{
border-image-source: url(border.png);
border-image-slice: 50% 50%;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Opera 不支持 border-image-slice 属性。

Internet Explorer 10 以及更早的版本不支持 border-image-slice 属性。

Safari 5 以及更早的版本不支持 border-image-slice 属性。

请参阅 border-image 属性。

定义和用法

border-image-slice 属性规定图像边框的向内偏移。

默认值：	100%
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.borderImageSlice="50% 50%"
---	---

语法

```
border-image-slice: _number_|_%_|fill;
```

注释：该属性规定图像的上、右、下、左侧边缘的向内偏移，图像被分割为九个区域：四个角、四条边以及一个中间区域。除非使用了关键词 fill，否则中间的图像部分会被丢弃。如果省略第四个数值/百分比，则与第二个值相同。如果省略第三个值，则与第一个值相同。如果省略第二个值，则与第一个值相同。

值	描述
number	数字值，代表图像中像素（如果是光栅图像）或矢量坐标（如果是矢量图像）。
%	相对于图像尺寸的百分比值：图像的宽度影响水平偏移，高度影响垂直偏移。
fill	保留边框图像的中间部分。

相关页面

CSS3 教程：CSS3 边框

CSS3 border-image-source 属性

实例

使用一幅图像作为围绕 div 元素的边框：

```
div
{
border-image-source: url(border.png);
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Opera 不支持 border-image-source 属性。

Internet Explorer 10 以及更早的版本不支持 border-image-source 属性。

Safari 5 以及更早的版本不支持 border-image-source 属性。

请参阅 border-image 属性。

定义和用法

border-image-source 属性规定要使用的图像，代替 border-style 属性中设置的边框样式。

提示：如果值为 "none"，或者如果图像无法显示，则使用边框样式。

默认值：	none
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderImageSource="url(border.png)"

语法

```
border-image-source: none|_image_;
```

值	描述
none	不使用图像。
image	用作边框的图像的路径。

相关页面

CSS3 教程：CSS3 边框

CSS3 border-image-width 属性

实例

规定图像边框的宽度：

```
div
{
border-image-source: url(border.png);
border-image-width: 30 30;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Opera 不支持 border-image-width 属性。

Internet Explorer 10 以及更早的版本不支持 border-image-width 属性。

Safari 5 以及更早的版本不支持 border-image-width 属性。

请参阅 border-image 属性。

定义和用法

border-image-width 属性规定图像边框的宽度。

默认值：	none
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderImageWidth="30 30"

语法

```
border-image-width: _number_|_%_|auto;
```

注释：border-image-width 属性的四个之规定将边框图像分割为九个部分的偏移。它们代表了从区域的上、右、下、左侧向内的距离。如果忽略第四个值，则与第二个值相同。如果省略第三个值，则与第一个值相同。如果省略第二个值，则与第一个值相同。不允许任何负值作为 border-image-width 值。

值	描述
number	代表对应的 border-width 的倍数。
%	参考边框图像区域的尺寸：区域的高度影响水平偏移，宽度影响垂直偏移。
auto	如果规定该属性，则宽度为对应的图像切片的固有宽度。

相关页面

CSS3 教程：CSS3 边框

CSS3 border-radius 属性

实例

向 div 元素添加圆角边框：

```
div
{
border:2px solid;
border-radius:25px;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-radius 属性。

定义和用法

border-radius 属性是一个简写属性，用于设置四个 border-*-radius 属性。

提示：该属性允许您为元素添加圆角边框！

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderRadius="5px"

语法

```
border-radius: _1-4 length_|_%_ / _1-4 length_|_%_;
```

注释：按此顺序设置每个 radii 的四个值。如果省略 bottom-left，则与 top-right 相同。如果省略 bottom-right，则与 top-left 相同。如果省略 top-right，则与 top-left 相同。

值	描述	测试
length	定义圆角的形状。	测试
%	以百分比定义圆角的形状。	测试

例子 1

```
border-radius:2em;
```

等价于：

```
border-top-left-radius:2em;
border-top-right-radius:2em;
border-bottom-right-radius:2em;
border-bottom-left-radius:2em;
```

例子 2

```
border-radius: 2em 1em 4em / 0.5em 3em;
```

等价于：

```
border-top-left-radius: 2em 0.5em;
border-top-right-radius: 1em 3em;
border-bottom-right-radius: 4em 0.5em;
border-bottom-left-radius: 1em 3em;
```

相关页面

CSS3 教程：CSS3 边框

CSS3 border-top-left-radius 属性

实例

向 div 元素的左上角添加圆角边框：

```
div
{
border:2px solid;
border-top-left-radius:2em;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-top-left-radius 属性。

定义和用法

border-top-left-radius 属性定义左上角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值：	0
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.borderTopLeftRadius="5px"
---	---

语法

```
border-bottom-right-radius: _length_|_%_ [_length_|_%_];
```

注释：border-top-left-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

值	描述	测试
length	定义左下角的形状。	测试
%	以百分比值定义左下角的形状。	测试

相关页面

CSS3 教程：CSS3 边框

CSS3 border-top-right-radius 属性

实例

向 div 元素的右上角添加圆角边框：

```
div
{
border:2px solid;
border-top-right-radius:2em;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-top-right-radius 属性。

定义和用法

border-top-right-radius 属性定义右下角边框的形状。

提示：该属性允许您向元素添加圆角边框。

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.borderTopRightRadius="5px"

语法

```
border-top-right-radius: _length_|_%_ [_length_|_%_];
```

注释：border-top-right-radius 属性的长度值和百分比值定义四分之一椭圆（定义外部边框边缘的边角形状）的半径（radii）。第一个值是水平半径，第二个值是垂直半径。如果省略第二个值，则复制第一个值。如果长度为零，则边角为方形，而不是圆形。水平半径的百分比值参考边框盒的宽度，而垂直半径的百分比值参考边框盒的高度。

值	描述	测试
length	定义右上角的形状。	测试
%	以百分比值定义右上角的形状。	测试

相关页面

CSS3 教程：CSS3 边框

CSS3 box-shadow 属性

实例

向 div 元素添加 box-shadow：

```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

IE9+、Firefox 4、Chrome、Opera 以及 Safari 5.1.1 支持 box-shadow 属性。

定义和用法

box-shadow 属性向框添加一个或多个阴影。

提示：请使用 border-image-* 属性来构造漂亮的可伸缩按钮！

默认值：	none
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.boxShadow="10px 10px 5px #888888"
---	---

语法

```
box-shadow: _h-shadow_ _v-shadow_ _blur_ _spread_ _color_ inset;
```

注释：box-shadow 向框添加一个或多个阴影。该属性是由逗号分隔的阴影列表，每个阴影由 2-4 个长度值、可选的颜色值以及可选的 inset 关键词来规定。省略长度的值是 0。

值	描述	测试
h-shadow	必需。水平阴影的位置。允许负值。	测试
v-shadow	必需。垂直阴影的位置。允许负值。	测试
blur	可选。模糊距离。	测试
spread	可选。阴影的尺寸。	测试
color	可选。阴影的颜色。请参阅 CSS 颜色值。	测试
inset	可选。将外部阴影 (outset) 改为内部阴影。	测试

亲自试一试 - 实例

扔到桌子上面的图片

本例演示如何创建“宝丽来”图片，并旋转图片。

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
margin:30px;
background-color:#E9E9E9;
}

div.polaroid
{
width:294px;
padding:10px 10px 20px 10px;
border:1px solid #BFBFBF;
background-color:white;
/* Add box-shadow */
box-shadow:2px 2px 3px #aaaaaa;
}

div.rotate_left
{
float:left;
-ms-transform:rotate(7deg); /* IE 9 */
```

```
-moz-transform:rotate(7deg); /* Firefox */
-webkit-transform:rotate(7deg); /* Safari and Chrome */
-o-transform:rotate(7deg); /* Opera */
transform:rotate(7deg);
}

div.rotate_right
{
float:left;
-ms-transform:rotate(-8deg); /* IE 9 */
-moz-transform:rotate(-8deg); /* Firefox */
-webkit-transform:rotate(-8deg); /* Safari and Chrome */
-o-transform:rotate(-8deg); /* Opera */
transform:rotate(-8deg);
}
</style>
</head>
<body>

<div class="polaroid rotate_left">
<img src="/i/ballade_dream.jpg" alt="郁金香" width="284" height="21
<p class="caption">上海鲜花港的郁金香，花名：Ballade Dream。</p>
</div>

<div class="polaroid rotate_right">
<img src="/i/china_pavilion.jpg" alt="世博中国馆" width="284" height
<p class="caption">2010年上海世博会，中国馆。</p>
</div>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 边框

Box 属性

属性	描述	CSS
overflow-x	如果内容溢出了元素内容区域，是否对内容的左/右边缘进行裁剪。	3
overflow-y	如果内容溢出了元素内容区域，是否对内容的上/下边缘进行裁剪。	3
overflow-style	规定溢出元素的首选滚动方法。	3
rotation	围绕由 rotation-point 属性定义的点对元素进行旋转。	3
rotation-point	定义距离上左边框边缘的偏移点。	3

CSS3 overflow-x 属性

实例

裁剪 div 元素中内容的左/右边缘 - 如果溢出元素的内容区域的话：

```
div
{
overflow-x:hidden;
}
```

浏览器支持

所有主流浏览器都支持 overflow-x 属性。

注释：overflow-x 属性无法在 IE8 以及更早的浏览器正确地工作。

定义和用法

overflow-x 属性规定是否对内容的左/右边缘进行裁剪 - 如果溢出元素内容区域的话。

提示：使用 overflow-y 属性来确定对上/下边缘的裁剪。

默认值：	visible
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.overflowX="scroll"

语法

```
overflow-x: visible|hidden|scroll|auto|no-display|no-content;
```

值	描述	测试
visible	不裁剪内容，可能会显示在内容框之外。	测试
hidden	裁剪内容 - 不提供滚动机制。	测试
scroll	裁剪内容 - 提供滚动机制。	测试
auto	如果溢出框，则应该提供滚动机制。	测试
no-display	如果内容不适合内容框，则删除整个框。	测试
no-content	如果内容不适合内容框，则隐藏整个内容。	测试

CSS3 overflow-y 属性

实例

裁剪 div 元素中内容的左/右边缘 - 如果溢出元素的内容区域的话：

```
div
{
overflow-y:hidden;
}
```

浏览器支持

所有主流浏览器都支持 overflow-y 属性。

注释：overflow-y 属性无法在 IE8 以及更早的浏览器正确地工作。

定义和用法

overflow-y 属性规定是否对内容的上/下边缘进行裁剪 - 如果溢出元素内容区域的话。

提示：使用 overflow-x 属性来确定对左/右边缘的裁剪。

默认值：	visible
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.overflowY="scroll"
---	---

语法

```
overflow-y: visible|hidden|scroll|auto|no-display|no-content;
```

值	描述	测试
visible	不裁剪内容，可能会显示在内容框之外。	测试
hidden	裁剪内容 - 不提供滚动机制。	测试
scroll	裁剪内容 - 提供滚动机制。	测试
auto	如果溢出框，则应该提供滚动机制。	测试
no-display	如果内容不适合内容框，则删除整个框。	测试
no-content	如果内容不适合内容框，则隐藏整个内容。	测试

CSS3 overflow-style 属性

实例

设置溢出元素的首选滚动方法：

```
div
{
overflow:auto;
overflow-style:marquee,panner;
}
```

浏览器支持

目前没有浏览器支持 overflow-style 属性。

定义和用法

overflow-style 规定溢出元素的首选滚动方法。

默认值：	**auto**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.overflowStyle="scrollbar"

语法

```
overflow-style: auto|scrollbar|panner|move|marquee;
```

注释：值既可以是 auto，或者是按照优先次序的方法列表。浏览器应该使用列表中其支持的的第一个滚动方法。

值	描述
auto	
scrollbar	为溢出元素添加滚动条。
panner	
move	用户能够直接移动元素的内容。通常，用户能够用鼠标拖动内容。
marquee	内容自主移动，不需任何用户代理对其控制。

CSS3 rotation 属性

实例

将 h1 元素旋转 180 度（从上向下）：

```
h1
{
rotation-point:50% 50%;
rotation:180deg;
}
```

浏览器支持

目前没有浏览器支持 rotation 属性。

定义和用法

rotation 属性围绕由 rotation-point 属性定义的指定点，对块级元素进行逆时针旋转。

提示：边框、内边距、内容以及背景（非固定）也会被旋转！

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.rotation="180deg"

语法

```
rotation: _angle_;
```

值	描述
angle	元素旋转角度。可能的值：0deg 到 360deg。

CSS3 rotation-point 属性

实例

将 h1 元素旋转 180 度（从上向下）：

```
h1
{
rotation-point:50% 50%;
rotation:180deg;
}
```

浏览器支持

目前没有浏览器支持 rotation-point 属性。

定义和用法

rotation-point 属性是一对值，定义从上左边框边缘进行偏移的点。

提示：rotation-point 属性需要与 rotation 属性结合使用。

默认值：	50% 50%
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.rotationPoint="25% 25%"

语法

```
rotation-point: _value_;
```

值	描述

- left top
- left center
- left bottom
- right top
- right center
- right bottom
- center top
- center center
- center bottom

| 如果只规定一个关键词，则第二个值将是 "center"。 | | *x% y%* | 百分比值，参考边框盒宽度和高度。 |

Color 属性

属性	描述	CSS
color-profile	允许使用源的颜色配置文件的默认以外的规范。	3
opacity	规定书签的级别。	3
rendering-intent	允许使用颜色配置文件渲染意图的默认以外的规范。	3

CSS3 opacity 属性

实例

设置 div 元素的不透明级别：

```
div
{
opacity:0.5;
}
```

您可以在本页底部找到更多实例。

浏览器支持

所有浏览器都支持 opacity 属性。

注释：IE8 以及更早的版本支持替代的 filter 属性。例如：
filter:Alpha(opacity=50)。

定义和用法

opacity 属性设置元素的不透明级别。

默认值：	**1**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.opacity=0.5

语法

```
opacity: _value_|inherit;
```

值	描述	测试
value	规定不透明度。从 0.0 （完全透明）到 1.0（完全不透明）。	测试
inherit	应该从父元素继承 opacity 属性的值。	

亲自试一试 - 实例

改变元素的不透明度

本例演示如何使用 JavaScript 来改变元素的不透明度。

```
<!DOCTYPE html>
<html>
<head>
<script>
function ChangeOpacity(x)
{
// 返回被选选项的文本
var opacity=x.options[x.selectedIndex].text;
var el=document.getElementById("p1");
if (el.style.opacity!==undefined)
  {el.style.opacity=opacity;}
else
  {alert("Your browser doesn't support this example!");}
}
</script>
</head>
<body>

<p id="p1">请从下面的例子中选择一个值，以改变此元素的不透明度。</p>
<select onchange="ChangeOpacity(this);" size="5">
  <option />0
  <option />0.2
  <option />0.5
  <option />0.8
  <option selected="selected" />1
</select>

</body>
</html>
```

相关页面

CSS 教程：CSS 图像透明度

Content for Paged Media 属性

属性	描述	CSS
bookmark-label	规定书签的标记。	3
bookmark-level	规定书签的级别。	3
bookmark-target	规定书签链接的目标。	3
float-offset	将元素放在 float 属性通常放置的位置的相反方向。	3
hyphenate-after	规定连字单词中连字符之后的最小字符数。	3
hyphenate-before	规定连字单词中连字符之前的最小字符数。	3
hyphenate-character	规定当发生断字时显示的字符串。	3
hyphenate-lines	指示元素中连续断字连线的最大数。	3
hyphenate-resource	规定帮助浏览器确定断字点的外部资源（逗号分隔的列表）。	3
hyphens	设置如何对单词进行拆分，以改善段落的布局。	3
image-resolution	规定图像的正确分辨率。	3
marks	向文档添加裁切标记或十字标记。	3
string-set	3	

CSS 尺寸属性（Dimension）

属性	描述	CSS
height	设置元素高度。	1
max-height	设置元素的最大高度。	2
max-width	设置元素的最大宽度。	2
min-height	设置元素的最小高度。	2
min-width	设置元素的最小宽度。	2
width	设置元素的宽度。	1

CSS height 属性

实例

设置段落的高度和宽度：

```
p
  {
  height:100px;
  width:100px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 height 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

height 属性设置元素的高度。

说明

这个属性定义元素内容区的高度，在内容区外面可以增加内边距、边框和外边距。

行内非替换元素会忽略这个属性。

默认值：	auto
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.height="50px"

可能的值

值	描述
auto	默认。浏览器会计算出实际的高度。
length	使用 px、cm 等单位定义高度。
%	基于包含它的块级对象的百分比高度。
inherit	规定应该从父元素继承 height 属性的值。

TIY 实例

使用像素值设置图像的高度

本例演示如何使用像素值设置元素的高度。

```
<html>
<head>
<style type="text/css">
img.normal
{
height: auto
}

img.big
{
height: 160px
}

img.small
{
height: 30px
}
</style>
</head>
<body>

<img class="normal" src="/i/eg_smile.gif" />
<br />
<img class="big" src="/i/eg_smile.gif" />
<br />
<img class="small" src="/i/eg_smile.gif" />

</body>
</html>
```

使用百分比设置图像的高度

本例演示如何使用百分比值来设置元素的高度。

```
<html>
<head>
<style type="text/css">
img.normal
{
height: auto
}

img.big
{
height: 50%
}

img.small
{
height: 10%
}
</style>
</head>
<body>

<img class="normal" src="/i/eg_smile.gif" />
<br />
<img class="big" src="/i/eg_smile.gif" />
<br />
<img class="small" src="/i/eg_smile.gif" />

</body>
</html>
```

相关页面

CSS 教程：CSS 尺寸

CSS 教程：CSS 框模型概述

CSS 参考手册：width 属性

HTML DOM 参考手册：height 属性

CSS max-height 属性

实例

设置段落的最大高度：

```
p
  {
  max-height:100px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 max-height 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

max-height 属性设置元素的最大高度。

说明

该属性值会对元素的高度设置一个最高限制。因此，元素可以比指定值矮，但不能比其高。不允许指定负值。

注释：max-height 属性不包括外边距、边框和内边距。

默认值：	none
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.maxHeight="50px"

可能的值

值	描述
none	默认。定义对元素被允许的最大高度没有限制。
length	定义元素的最大高度值。
%	定义基于包含它的块级对象的百分比最大高度。
inherit	规定应该从父元素继承 max-height 属性的值。

TIY 实例

设置元素的最大高度

本例演示如何设置一个元素的最大高度。

```
<html>
<head>
<style type="text/css">
p
{
max-height: 10px
}
</style>
</head>
<body>

<p>这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg_smile.gif" />

</body>
</html>
```

相关页面

CSS 教程：CSS 尺寸

CSS 教程：min-height 属性

HTML DOM 参考手册：maxHeight 属性

CSS max-width 属性

实例

设置段落的最大宽度：

```
p
  {
  max-width:100px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 max-width 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

max-width 定义元素的最大宽度。

说明

该属性值会对元素的宽度设置一个最高限制。因此，元素可以比指定值窄，但不能比其宽。不允许指定负值。

默认值：	none
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.maxWidth="50px"
---	---

可能的值

值	描述
none	默认。定义对元素的最大宽度没有限制。
length	定义元素的最大宽度值。
%	定义基于包含它的块级对象的百分比最大宽度。
inherit	规定应该从父元素继承 max-width 属性的值。

TIY 实例

使用像素值来设置元素的最大宽度

本例演示如何使用像素值来设置元素的最大高度。

```
<html>
<head>
<style type="text/css">
p
{
max-width: 300px
}
</style>
</head>
<body>

<p>这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。</p>

</body>
</html>
```

使用百分比来设置元素的最大宽度

本例演示如何使用百分比值来设置元素的最大高度。

```
<html>
<head>
<style type="text/css">
p
{
max-width: 50%
}
</style>
</head>
<body>

<p>这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 尺寸

CSS 参考手册：CSS min-width 属性

HTML DOM 参考手册：maxWidth 属性

CSS min-height 属性

实例

设置段落的最小高度：

```
p
  {
  min-height:100px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 min-height 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

min-height 属性设置元素的最小高度。

说明

该属性值会对元素的高度设置一个最低限制。因此，元素可以比指定值高，但不能比其矮。不允许指定负值。

默认值：	0
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.minHeight="50px"
---	---

可能的值

值	描述
length	定义元素的最小高度。默认值是 0。
%	定义基于包含它的块级对象的百分比最小高度。
inherit	规定应该从父元素继承 min-height 属性的值。

TIY 实例

使用像素值来设置元素的最小高度

本例演示如何使用像素值来设置元素的最小高度。

```
<html>
<head>
<style type="text/css">
p
{
min-height: 100px
}
</style>
</head>
<body>

<p>这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg_smile.gif" />

</body>
</html>
```

相关页面

CSS 教程：CSS 尺寸

CSS 教程：max-height 属性

HTML DOM 参考手册：minHeight 属性

CSS min-width 属性

实例

设置段落的最小宽度：

```
p
  {
  min-width:100px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 min-width 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

min-width 属性设置元素的最小宽度。

说明

该属性值会对元素的宽度设置一个最小限制。因此，元素可以比指定值宽，但不能比其窄。不允许指定负值。

默认值：	none
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.minWidth="50px"

可能的值

值	描述
length	定义元素的最小宽度值。默认值：取决于浏览器。
%	定义基于包含它的块级对象的百分比最小宽度。
inherit	规定应该从父元素继承 min-width 属性的值。

TIY 实例

使用像素值来设置元素的最小宽度

本例演示如何使用像素值来设置元素的最小宽度。

```
<html>
<head>
<style type="text/css">
p
{
min-width: 1000px
}
</style>
</head>
<body>

<p>这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg_smile.gif" />

</body>
</html>
```

使用百分比来设置元素的最小宽度

本例演示如何使用百分比值来设置元素的最小宽度。

```
<html>
<head>
<style type="text/css">
p
{
min-width: 200%
}
</style>
</head>
<body>

<p>这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。</p>

<img src="/i/eg_smile.gif" />

</body>
</html>
```

相关页面

CSS 教程：CSS 尺寸

CSS 参考手册：CSS max-width 属性

HTML DOM 参考手册：minWidth 属性

CSS width 属性

实例

设置段落的高度和宽度：

```
p
  {
  height:100px;
  width:100px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 width 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

width 属性设置元素的宽度。

说明

这个属性定义元素内容区的宽度，在内容区外面可以增加内边距、边框和外边距。

行内非替换元素会忽略这个属性。

默认值：	auto
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.width="50px"

可能的值

值	描述
auto	默认值。浏览器可计算出实际的宽度。
length	使用 px、cm 等单位定义宽度。
%	定义基于包含块（父元素）宽度的百分比宽度。
inherit	规定应该从父元素继承 width 属性的值。

TIY 实例

使用像素值来设置元素的宽度

本例演示如何使用像素值来设置元素的宽度。

```
<html>
<head>
<style type="text/css">
img
{
width: 300px
}
</style>
</head>
<body>

<img src="/i/eg_smile.gif" />

</body>
</html>
```

使用百分比来设置元素的宽度

本例演示如何使用百分比值来设置元素的宽度。

```
<html>
<head>
<style type="text/css">
img
{
width: 50%
}
</style>
</head>
<body>

<img src="/i/eg_smile.gif" />

</body>
</html>
```

相关页面

CSS 教程：CSS 尺寸

CSS 教程：CSS 框模型概述

CSS 参考手册：height 属性

HTML DOM 参考手册：width 属性

可伸缩框属性（Flexible Box）

属性	描述	CSS
box-align	规定如何对齐框的子元素。	3
box-direction	规定框的子元素的显示方向。	3
box-flex	规定框的子元素是否可伸缩。	3
box-flex-group	将可伸缩元素分配到柔性分组。	3
box-lines	规定当超出父元素框的空间时，是否换行显示。	3
box-ordinal-group	规定框的子元素的显示次序。	3
box-orient	规定框的子元素是否应水平或垂直排列。	3
box-pack	规定水平框中的水平位置或者垂直框中的垂直位置。	3

CSS3 box-align 属性

实例

通过使用 box-align and box-pack 属性，居中 div 框的子元素：

```
div
{
width:350px;
height:100px;
border:1px solid black;

/* Firefox */
display:-moz-box;
-moz-box-pack:center;
-moz-box-align:center;

/* Safari、Opera 以及 Chrome */
display:-webkit-box;
-webkit-box-pack:center;
-webkit-box-align:center;

/* W3C */
display:box;
box-pack:center;
box-align:center;
}
```

页面底部有更多实例。

浏览器支持

目前没有浏览器支持 box-align 属性。

Firefox 支持替代的 -moz-box-align 属性。

Safari、Opera 以及 Chrome 支持替代的 -webkit-box-align 属性。

定义和用法

box-align 属性规定如何对齐框的子元素。

默认值：	stretch
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxAlign="center"

语法

```
box-align: start|end|center|baseline|stretch;
```

值	描述	测试
start		

对于正常方向的框，每个子元素的上边缘沿着框的顶边放置。

对于反方向的框，每个子元素的下边缘沿着框的底边放置。

| 测试 | | end |

对于正常方向的框，每个子元素的下边缘沿着框的底边放置。

对于反方向的框，每个子元素的上边缘沿着框的顶边放置。

| 测试 | | center | 均等地分割多余的空间，一半位于子元素之上，另一半位于子元素之下。 | 测试 | | baseline | 如果 box-orient 是inline-axis或horizontal，所有子元素均与其基线对齐。 | 测试 | | stretch | 拉伸子元素以填充包含块 |

亲自试一试 - 实例

改变元素的 box-align 值

该例演示如何使用 JavaScript 来改变元素的 box-align 值。

```
<!DOCTYPE html>
<html>
<head>
<style>
.box
{
```

```
display:-moz-box;
display:-webkit-box;
display:box;
width:200px;
height:100px;
border:2px solid red;
}
</style>
<script>
function ChangeBoxAlign(x)
{
// Returns the selected option's text
var boxAlign=x.options[x.selectedIndex].text;
var div=document.getElementById("myDiv");
if (div.style.MozBoxAlign!==undefined)
  {
  div.style.MozBoxAlign=boxAlign;
  }
else if (div.style.webkitBoxAlign!==undefined)
  {
  div.style.webkitBoxAlign=boxAlign;
  }
else
  {
  alert("Your browser doesn't support this example!");
  }
}
</script>
</head>
<body>
<div class="box" id="myDiv">
  <b>第一个子元素</b>
  <i>第二个子元素</i>
</div>

<select onchange="ChangeBoxAlign (this);" size="6">
  <option selected="selected" />baseline
  <option />center
  <option />end
  <option />inherit
  <option />start
  <option />stretch
</select>

<p><b>注释：</b>本例在 Internet Explorer 和 Opera 中无效。</p>

</body>
</html>
```

CSS3 box-direction 属性

实例

以反方向显示 div 框的子元素：

```
div
{
width:350px;
height:100px;
border:1px solid black;

/* Firefox */
display:-moz-box;
-moz-box-direction:reverse;

/* Safari、Opera 以及 Chrome */
display:-webkit-box;
-webkit-box-direction:reverse;

/* W3C */
display:box;
box-direction:reverse;
}
```

浏览器支持

目前没有浏览器支持 box-direction 属性。

Firefox 支持替代的 -moz-box-direction 属性。

Safari、Opera 以及 Chrome 支持替代的 -webkit-box-direction 属性。

定义和用法

box-direction 属性规定框元素的子元素以什么方向来排列。

默认值：	**normal**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxDirection="reverse"

语法

```
box-direction: normal|reverse|inherit;
```

值	描述	测试
normal	以默认方向显示子元素。	测试
reverse	以反方向显示子元素。	测试
inherit	应该从子元素继承 box-direction 属性的值	

CSS3 box-flex 属性

实例

定义两个可伸缩的 p 元素。如果父元素的总宽度是 300 像素，则 #p1 的宽度是 100 像素，而 #p2 的宽度是 200 像素：

```
#p1
{
-moz-box-flex:1.0; /* Firefox */
-webkit-box-flex:1.0; /* Safari 和 Chrome */
box-flex:1.0;
border:1px solid red;
}

#p2
{
-moz-box-flex:2.0; /* Firefox */
-webkit-box-flex:2.0; /* Safari 和 Chrome */
box-flex:2.0;
border:1px solid blue;
}
```

浏览器支持

目前没有浏览器支持 box-flex 属性。

Firefox 支持替代的 -moz-box-flex 属性。

Safari、Opera 以及 Chrome 支持替代的 -webkit-box-flex 属性。

定义和用法

box-flex 属性规定框的子元素是否可伸缩其尺寸。

提示：可伸缩元素能够随着框的缩小或扩大而缩写或放大。只要框中有多余的空间，可伸缩元素就会扩展来填充这些空间。

默认值：	**0.0**（指示该元素不可伸缩）
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxFlex=2.0

语法

```
box-flex: _value_;
```

值	描述
value	元素的可伸缩行。柔性是相对的，例如 box-flex 为 2 的子元素两倍于 box-flex 为 1 的子元素。

CSS3 box-flex-group 属性

浏览器支持

目前没有浏览器支持 box-flex-group 属性。

定义和用法

box-flex-group 属性用于向柔性分组分配可伸缩元素。

提示：可伸缩元素能够随着框的缩小和扩大而伸缩。

默认值：	1
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxFlexGroup=2

语法

```
box-flex-group: _integer_;
```

值	描述
integer	一个整数。（第一个柔性分组是 1，后面的柔性分组是更大的值）。

CSS3 box-lines 属性

实例

规定允许 div 扩展为多行：

```
div
{
display:box;
box-orient:horizontal;
box-lines:multiple;
width:200px;
}
```

浏览器支持

目前没有浏览器支持 box-lines 属性。

定义和用法

box-lines 属性规定如果列超出了父框中的空间，是否要换行显示。

提示：默认地，水平框会在单独的行中排列其子元素，而垂直框会在单独的列中排列其子元素。

默认值：	single
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxLines="multiple"

语法

```
box-lines: single|multiple;
```

值	描述
single	所有子元素会被放置在单独的行或列中。（无法匹配的元素会被视为溢出）。
multiple	允许框扩展为多行，以容纳其所有子元素。

CSS3 box-ordinal-group 属性

实例

规定框中子元素的显示次序：

```
.box
{
display:-moz-box; /* Firefox */
display:-webkit-box; /* Safari 和 Chrome */
display:box;
border:1px solid black;
}
.ord1
{
margin:5px;
-moz-box-ordinal-group:1; /* Firefox */
-webkit-box-ordinal-group:1; /* Safari 和 Chrome */
box-ordinal-group:1;
}
.ord2
{
margin:5px;
-moz-box-ordinal-group:2; /* Firefox */
-webkit-box-ordinal-group:2; /* Safari 和 Chrome */
box-ordinal-group:2;
}
```

浏览器支持

目前没有浏览器支持 box-ordinal-group 属性。

Firefox 支持替代的 -moz-box-ordinal-group 属性。

Safari 和 Chrome 支持替代的 -webkit-box-ordinal-group 属性。

定义和用法

box-ordinal-group 属性规定框中子元素的显示次序。

值更低的元素会显示在值更高的元素前面显示。

注释：分组值相同的元素，它们的显示次序取决于其源次序。

默认值：	**1**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxOrdinalGroup=2

语法

```
box-ordinal-group: _integer_;
```

值	描述	测试
integer	一个整数，指示子元素的显示次序。	测试

CSS3 box-orient 属性

实例

水平排列 div 元素的子元素：

```
div
{
width:350px;
height:100px;
border:1px solid black;

/* Firefox */
display:-moz-box;
-moz-box-orient:horizontal;

/* Safari、Opera 以及 Chrome */
display:-webkit-box;
-webkit-box-orient:horizontal;

/* W3C */
display:box;
box-orient:horizontal;
}
```

浏览器支持

目前没有浏览器支持 box-orient 属性。

Firefox 支持替代的 -moz-box-orient 属性。

Safari、Opera 以及 Chrome 支持替代的 -webkit-box-orient 属性。

定义和用法

box-orient 属性规定框的子元素应该被水平或垂直排列。

提示：水平框中的子元素从左向右进行显示，而垂直框的子元素从上向下进行显示。不过，box-direction 和 box-ordinal-group 能够改变这种顺序。

默认值：	**inline-axis**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxOrient="vertical"

语法

```
box-orient: horizontal|vertical|inline-axis|block-axis|inherit;
```

值	描述	测试
horizontal	在水平行中从左向右排列子元素。	测试
vertical	从上向下垂直排列子元素。	测试
inline-axis	沿着行内轴来排列子元素（映射为 horizontal）。	测试
block-axis	沿着块轴来排列子元素（映射为 vertical）。	测试
inherit	应该从父元素继承 box-orient 属性的值。	

CSS3 box-pack 属性

实例

通过一起使用 box-align 和 box-pack 属性，对 div 框的子元素进行居中：

```
div
{
width:350px;
height:100px;
border:1px solid black;

/* Firefox */
display:-moz-box;
-moz-box-pack:center;
-moz-box-align:center;

/* Safari、Opera 以及 Chrome */
display:-webkit-box;
-webkit-box-pack:center;
-webkit-box-align:center;

/* W3C */
display:box;
box-pack:center;
box-align:center;
}
```

浏览器支持

目前没有浏览器支持 box-pack 属性。

Firefox 支持替代的 -moz-box-pack 属性。

Safari、Opera 以及 Chrome 支持替代的 -webkit-box-pack 属性。

定义和用法

box-pack 属性规定当框大于子元素的尺寸，在何处放置子元素。

该属性规定水平框中的水平位置，以及垂直框中的垂直位置。

默认值：	start
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxPack="center"

语法

```
box-pack: start|end|center|justify;
```

值	描述	测试
start		

对于正常方向的框，首个子元素的左边缘被放在左侧（最后的子元素后是所有剩余的空间）

对于相反方向的框，最后子元素的右边缘被放在右侧（首个子元素前是所有剩余的空间）

| 测试 | | end |

对于正常方向的框，最后子元素的右边缘被放在右侧（首个子元素前是所有剩余的空间）。

对于相反方向的框，首个子元素的左边缘被放在左侧（最后子元素后是所有剩余的空间）。 | 测试 | | center | 均等地分割多余空间，其中一半空间被置于首个子元素前，另一半被置于最后一个子元素后 | 测试 | | justify | 在每个子元素之间分割多余的空间（首个子元素前和最后一个子元素后没有多余的空间）。 | 测试 |

CSS 字体属性（Font）

属性	描述	CSS
font	在一个声明中设置所有字体属性。	1
font-family	规定文本的字体系列。	1
font-size	规定文本的字体尺寸。	1
font-size-adjust	为元素规定 aspect 值。	2
font-stretch	收缩或拉伸当前的字体系列。	2
font-style	规定文本的字体样式。	1
font-variant	规定是否以小型大写字母的字体显示文本。	1
font-weight	规定字体的粗细。	1

CSS font 属性

实例

在一个声明中设置所有字体属性：

```
p.ex1
  {
  font:italic arial,sans-serif;
  }

p.ex2
  {
  font:italic bold 12px/20px arial,sans-serif;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 font 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

font 简写属性在一个声明中设置所有字体属性。

注释：此属性也有第六个值："line-height"，可设置行间距。

说明

这个简写属性用于一次设置元素字体的两个或更多方面。使用 icon 等关键字可以适当地设置元素的字体，使之与用户计算机环境中的某个方面一致。注意，如果没有使用这些关键词，至少要指定字体大小和字体系列。

可以按顺序设置如下属性：

- font-style
- font-variant
- font-weight
- font-size/line-height

- font-family

可以不设置其中的某个值，比如 font:100% verdana; 也是允许的。未设置的属性会使用其默认值。

默认值：	***not specified***
继承性：	yes
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.font="italic small-caps bold 12px arial,sans-serif"
---	---

可能的值

值	描述
font-style	规定字体样式。参阅：font-style 中可能的值。
font-variant	规定字体异体。参阅：font-variant 中可能的值。
font-weight	规定字体粗细。参阅：font-weight 中可能的值。
font-size/line-height	规定字体尺寸和行高。参阅：font-size 和 line-height 中可能的值。
font-family	规定字体系列。参阅：font-family 中可能的值。
caption	定义被标题控件（比如按钮、下拉列表等）使用的字体。
icon	定义被图标标记使用的字体。
menu	定义被下拉列表使用的字体。
message-box	定义被对话框使用的字体。
small-caption	caption 字体的小型版本。
status-bar	定义被窗口状态栏使用的字体。

TIY 实例

所有字体属性在一个声明之内

本例演示如何使用简写属性将字体属性设置在一个声明之内。

```
<html>
<head>
<style type="text/css">
p.ex1
{
font:italic arial,sans-serif;
}

p.ex2
{
font:italic bold 12px/30px arial,sans-serif;
}
</style>
</head>

<body>
<p class="ex1">This is a paragraph. This is a paragraph. This is a
<p class="ex2">This is a paragraph. This is a paragraph. This is a
</body>
</html>
```

设置使用 "caption" 值的段落字体

本例演示如何设置使用 "caption" 值的段落字体。

```
<html>
<body>

<p>This is a normal paragraph</p>
<p style="font: caption">This is a paragraph with a "caption" font<

</body>
</html>
```

相关页面

CSS 教程：CSS 字体

HTML DOM 参考手册：font 属性

CSS font-family 属性

实例

为段落设置字体：

```
p
  {
  font-family:"Times New Roman",Georgia,Serif;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 font-family 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

font-family 规定元素的字体系列。

font-family 可以把多个字体名称作为一个“回退”系统来保存。如果浏览器不支持第一个字体，则会尝试下一个。也就是说，font-family 属性的值是用于某个元素的字体族名称或/及类族名称的一个优先表。浏览器会使用它可识别的第一个值。

有两种类型的字体系列名称：

- 指定的系列名称：具体字体的名称，比如："times"、"courier"、"arial"。
- 通常字体系列名称：比如："serif"、"sans-serif"、"cursive"、"fantasy"、"monospace"

提示：使用逗号分割每个值，并始终提供一个类族名称作为最后的选择。

注意：使用某种特定的字体系列（Geneva）完全取决于用户机器上该字体系列是否可用；这个属性没有指示任何字体下载。因此，强烈推荐使用一个通用字体系列名作为后路。

默认值：	***not specified***
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.fontFamily="arial,sans-serif"

可能的值

值	描述

- *family-name*
- *generic-family*

|

用于某个元素的字体族名称或/及类族名称的一个优先表。

默认值：取决于浏览器。

| | inherit | 规定应该从父元素继承字体系列。 |

TIY 实例

设置文本的字体

本例演示如何设置文本字体。

```
<html>
<head>
<style type="text/css">
p.serif{font-family:"Times New Roman",Georgia,Serif}
p.sansserif{font-family:Arial,Verdana,Sans-serif}
</style>
</head>

<body>
<h1>CSS font-family</h1>
<p class="serif">This is a paragraph, shown in the Times New Roman
<p class="sansserif">This is a paragraph, shown in the Arial font.<

</body>
</html>
```

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：font 属性

CSS font-size 属性

实例

设置不同的 HTML 元素的尺寸：

```
h1 {font-size:250%;}
h2 {font-size:200%;}
p {font-size:100%}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 font-size 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

font-size 属性可设置字体的尺寸。

说明

该属性设置元素的字体大小。注意，实际上它设置的是字体中字符框的高度；实际的字符字形可能比这些框高或矮（通常会矮）。

各关键字对应的字体必须比一个最小关键字相应字体要高，并且要小于下一个最大关键字对应的字体。

默认值：	**medium**
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.fontSize="larger"

可能的值

值	描述

- xx-small
- x-small
- small
- medium
- large
- x-large
- xx-large

  |

把字体的尺寸设置为不同的尺寸，从 xx-small 到 xx-large。

默认值：medium。

| | smaller | 把 font-size 设置为比父元素更小的尺寸。 | | larger | 把 font-size 设置为比父元素更大的尺寸。 | | *length* | 把 font-size 设置为一个固定的值。 | | % | 把 font-size 设置为基于父元素的一个百分比值。 | | inherit | 规定应该从父元素继承字体尺寸。 |

TIY 实例

设置字体尺寸

本例演示如何设置字体尺寸。

```
<html>
<head>
<style type="text/css">
h1 {font-size: 300%}
h2 {font-size: 200%}
p {font-size: 100%}
</style>
</head>

<body>
<h1>This is header 1</h1>
<h2>This is header 2</h2>
<p>This is a paragraph</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：fontSize 属性

CSS font-size-adjust 属性

实例

设置不同的 HTML 元素的 font-size-adjust 属性：

```
h1
  {
  font-size-adjust:0.58;
  }

p
  {
  font-size-adjust:0.60;
  }
```

浏览器支持

Internet Explorer 不支持 font-size-adjust 属性。

定义和用法

font-size-adjust 属性为某个元素规定一个 aspect 值，这样就可以保持首选字体的 x-height。

说明

字体的小写字母 "x" 的高度与 "font-size" 高度之间的比率被称为一个字体的 aspect 值。当字体拥有高的 aspect 值时，那么当此字体被设置为很小的尺寸时会更易阅读。举例：Verdana 的 aspect 值是 0.58（意味着当字体尺寸为 100px 时，它的 x-height 是 58px ）。Times New Roman 的 aspect 值是 0.46。这就意味着 Verdana 在小尺寸时比 Times New Roman 更易阅读。

默认值：	**none**
继承性：	yes
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.fontSizeAdjust="0.70"
---	---

可能的值

值	描述
none	默认。如果此字体不可用，则不保持此字体的 x-height。
number	

定义字体的 aspect 值比率。

可使用的公式：

首选字体的字体尺寸 * （font-size-adjust 值 / 可用字体的 aspect 值）=可应用到可用字体的字体尺寸

举例：

如果 14px 的 Verdana（aspect 值是 0.58）不可用，但是某个可用的字体的 aspect 值是 0.46，那么替代字体的尺寸将是 14 * (0.58/0.46) = 17.65px。

|

TIY 实例

使用 font-size-adjust 设置字体尺寸

本例演示如何使用 font-size-adjust 设置字体尺寸。

```
<html>
<head>
<style type="text/css">
h1 {font-size-adjust: 0.50}
h2 {font-size-adjust: 0.40}
p {font-size-adjust: 0.60}
</style>
</head>
<body>

<h1>This is header 1</h1>
<h2>This is header 2</h2>
<p>This is a paragraph</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：fontSizeAdjust 属性

CSS font-stretch 属性

实例

设置 HTML 元素的 font-stretch 属性：

```
h1
  {
  font-stretch:ultra-condensed;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都不支持 font-stretch 属性。

定义和用法

font-stretch 属性可对当前的 font-family 进行伸缩变形。

默认值：	normal
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.fontStretch="ultra-expanded"

可能的值

值	描述
normal	默认值。把缩放比例设置为标准。
wider	把伸展比例设置为更进一步的伸展值
narrower	把收缩比例设置为更进一步的收缩值

- ultra-condensed
- extra-condensed
- condensed
- semi-condensed
- semi-expanded
- expanded
- extra-expanded
- ultra-expanded

|

设置 font-family 的缩放比例。

"ultra-condensed" 是最宽的值，而 "ultra-expanded" 是最窄的值。

|

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：fontStretch 属性

CSS font-style 属性

实例

为三个段落设置不同的字体样式：

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 font-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

font-style 属性定义字体的风格。

说明

该属性设置使用斜体、倾斜或正常字体。斜体字体通常定义为字体系列中的一个单独的字体。理论上讲，用户代理可以根据正常字体计算一个斜体字体。

默认值：	normal
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.fontStyle="italic"

可能的值

值	描述
normal	默认值。浏览器显示一个标准的字体样式。
italic	浏览器会显示一个斜体的字体样式。
oblique	浏览器会显示一个倾斜的字体样式。
inherit	规定应该从父元素继承字体样式。

TIY 实例

设置字体风格

本例演示如何设置字体风格。

```
<html>
<head>
<style type="text/css">
p.normal {font-style:normal}
p.italic {font-style:italic}
p.oblique {font-style:oblique}
</style>
</head>

<body>
<p class="normal">This is a paragraph, normal.</p>
<p class="italic">This is a paragraph, italic.</p>
<p class="oblique">This is a paragraph, oblique.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：fontStyle 属性

CSS font-variant 属性

实例

把段落设置为小型大写字母字体：

```
p.small
  {
  font-variant:small-caps;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 font-variant 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

font-variant 属性设置小型大写字母的字体显示文本，这意味着所有的小写字母均会被转换为大写，但是所有使用小型大写字体的字母与其余文本相比，其字体尺寸更小。

说明

该属性主要用于定义小型大写字母文本。理论上，用户代理可以根据正常字体计算出小型大写字母字体。

默认值：	normal
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.fontVariant="small-caps"

可能的值

值	描述
normal	默认值。浏览器会显示一个标准的字体。
small-caps	浏览器会显示小型大写字母的字体。
inherit	规定应该从父元素继承 font-variant 属性的值。

TIY 实例

设置字体的异体

本例演示如何设置字体的异体。

```
<html>
<head>
<style type="text/css">
p.normal {font-variant: normal}
p.small {font-variant: small-caps}
</style>
</head>

<body>
<p class="normal">This is a paragraph</p>
<p class="small">This is a paragraph</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：fontVariant 属性

CSS font-weight 属性

实例

设置三个段落的字体的粗细：

```
p.normal {font-weight:normal;}
p.thick {font-weight:bold;}
p.thicker {font-weight:900;}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 font-weight 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

font-weight 属性设置文本的粗细。

说明

该属性用于设置显示元素的文本中所用的字体加粗。数字值 400 相当于 关键字 normal，700 等价于 bold。每个数字值对应的字体加粗必须至少与下一个最小数字一样细，而且至少与下一个最大数字一样粗。

默认值：	normal
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.fontWeight="900"

可能的值

值	描述
normal	默认值。定义标准的字符。
bold	定义粗体字符。
bolder	定义更粗的字符。
lighter	定义更细的字符。

- 100
- 200
- 300
- 400
- 500
- 600
- 700
- 800
- 900

  | 定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。 | | inherit | 规定应该从父元素继承字体的粗细。 |

TIY 实例

设置字体的粗细

本例演示如何设置字体的粗细。

```
<html>
<head>
<style type="text/css">
p.normal {font-weight: normal}
p.thick {font-weight: bold}
p.thicker {font-weight: 900}
</style>
</head>

<body>
<p class="normal">This is a paragraph</p>

<p class="thick">This is a paragraph</p>

<p class="thicker">This is a paragraph</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 字体

CSS 参考手册：CSS font 属性

HTML DOM 参考手册：fontWeight 属性

内容生成（Generated Content）

属性	描述	CSS
content	与 :before 以及 :after 伪元素配合使用，来插入生成内容。	2
counter-increment	递增或递减一个或多个计数器。	2
counter-reset	创建或重置一个或多个计数器。	2
quotes	设置嵌套引用的引号类型。	2
crop	允许被替换元素仅仅是对象的矩形区域，而不是整个对象。	3
move-to	从流中删除元素，然后在文档中后面的点上重新插入。	3
page-policy	确定元素基于页面的 occurrence 应用于计数器还是字符串值。	3

CSS content 属性

实例

下面的例子在每个链接后插入括号中的 URL：

```
a:after
  {
  content: " (" attr(href) ")";
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 content 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 content 属性。

定义和用法

content 属性与 :before 及 :after 伪元素配合使用，来插入生成内容。

说明

该属性用于定义元素之前或之后放置的生成内容。默认地，这往往是行内内容，不过该内容创建的框类型可以用属性 display 控制。

默认值：	**normal**
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.content="url(beep.wav)"

可能的值

值	描述
none	
normal	
content specifications	
inherit	规定应该从父元素继承 content 属性的值。

相关页面

CSS 参考手册：CSS :before 伪元素

CSS 参考手册：CSS :after 伪元素

HTML DOM 参考手册：content 属性

CSS counter-increment 属性

实例

对部分和子部分进行编号（比如 "Section 1"、"1.1"、"1.2"）的方法：

```
body
  {
  counter-reset:section;
  }

h1
  {
  counter-reset:subsection;
  }

h1:before
  {
  content:"Section " counter(section) ". ";
  counter-increment:section;
  }

h2:before
  {
  counter-increment:subsection;
  content:counter(section) "." counter(subsection) " ";
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 counter-increment 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 counter-increment 属性。

定义和用法

counter-increment 属性设置某个选取器每次出现的计数器增量。默认增量是 1。

说明

利用这个属性，计数器可以递增（或递减）某个值，这可以是正值或负值。如果没有提供 number 值，则默认为 1。

注释：如果使用了 "display: none"，则无法增加计数。如使用 "visibility: hidden"，则可增加计数。

默认值：	**none**
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.counterIncrement="subsection"
---	---

可能的值

值	描述
none	默认。选择器无计数器增量。
id number	

id 定义将增加计数的选择器、id 或 class。

number 定义增量。number 可以是正数、零或者负数。

| | inherit | 规定应该从父元素继承 counter-increment 属性的值。 |

相关页面

CSS 参考手册：CSS :before 伪元素

CSS 参考手册：CSS :after 伪元素

CSS 参考手册：content 属性

CSS 参考手册：counter-reset 属性

CSS counter-reset 属性

实例

对部分和子部分进行编号（比如 "Section 1"、"1.1"、"1.2"）的方法：

```
body
  {
  counter-reset:section;
  }

h1
  {
  counter-reset:subsection;
  }

h1:before
  {
  content:"Section " counter(section) ". ";
  counter-increment:section;
  }

h2:before
  {
  counter-increment:subsection;
  content:counter(section) "." counter(subsection) " ";
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 counter-reset 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 counter-reset 属性。

定义和用法

counter-reset 属性设置某个选择器出现次数的计数器的值。默认为 0。

利用这个属性，计数器可以设置或重置为任何值，可以是正值或负值。如果没有提供 number，则默认为 0。

注释：如果使用 "display: none"，则无法重置计数器。如果使用 "visibility: hidden"，则可以重置计数器。

默认值：	**none**
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.counterReset="subsection"

可能的值

值	描述
none	默认。不能对选择器的计数器进行重置。
id number	

id 定义重置计数器的选择器、id 或 class。

number 可设置此选择器出现次数的计数器的值。可以是正数、零或负数。

| | inherit | 规定应该从父元素继承 counter-reset 属性的值。 |

相关页面

CSS 参考手册：CSS :before 伪元素

CSS 参考手册：CSS :after 伪元素

CSS 参考手册：content 属性

CSS 参考手册：counter-increment 属性

CSS quotes 属性

实例

```
q:lang(en)
  {
  quotes: '"' '"' "'" "'";
  }
```

HTML 代码：

```
<html lang="en">
<head>
</head>
<body>
<p><q>This is a <q>big</q> quote</q></p>
</body>
</html>
```

输出：

```
"This is a 'big' quote"
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 quotes 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 quotes 属性。

定义和用法

quotes 属性设置嵌套引用（embedded quotation）的引号类型。

默认值：	**not specified**
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.quotes="none"

可能的值

值	描述
none	规定 "content" 属性的 "open-quote" 和 "close-quote" 的值不会产生任何引号。
string string string string	

定义要使用的引号。

前两个值规定第一级引用嵌套，后两个值规定下一级引号嵌套。

| | inherit | 规定应该从父元素继承 quotes 属性的值。 |

引号字符

Result	Description	Entity Number
"	double quote	"
'	single quote	'
‹	single, left angle quote	‹
›	single, right angle quote	›
«	double, left angle quote	«
»	double, right angle quote	»
‘	left quote (single high-6)	‘
’	right quote (single high-9)	’
“	left quote (double high-6)	“
”	right quote (double high-9)	”
„	double quote (double low-9)	„

相关页面

HTML DOM 参考手册：quotes 属性

Grid 属性

属性	描述	CSS
grid-columns	规定网格中每个列的宽度。	3
grid-rows	规定网格中每个列的高度。	3

CSS3 grid-columns 属性

实例

在 div 元素中部添加一个网格行，距右侧 200 像素处添加另一个，在余下空间的中间再添加一个：

```
div
{
grid-columns:50% * * 200px;
}
```

浏览器支持

目前没有浏览器支持 grid-columns。

定义和用法

grid-columns 属性网格中每列的宽度。

提示：使用网格系统对于打印设计师来说具有巨大的价值。现在相同的改变被应用到在线内容上。网格属性提供了在可伸缩网格中调整标题、文本和图片尺寸和位置的能力。

默认值：	**none**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.gridColumns="50% 200px"

语法

```
grid-columns: _length_|_%_|none|inherit;
```

值	描述
length	参考包含块的网格。
%	参考包含块的宽度。
none	
inherit	

CSS3 grid-rows 属性

实例

定义 100 像素的标题行，并按需添加多个额外的行，高度交替为 30 和 60 像素：

```
div
{
grid-rows:100px (30px 60px);
}
```

浏览器支持

目前没有浏览器支持 grid-rows。

定义和用法

grid-rows 属性规定网格中每行的高度。

提示：使用网格系统对于打印设计师来说具有巨大的价值。现在相同的改变被应用到在线内容上。网格属性提供了在可伸缩网格中调整标题、文本和图片尺寸和位置的能力。

默认值：	none
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.gridRows="100px (30px 60px)"

语法

```
grid-rows: _length_|_%_|none|inherit;
```

值	描述
length	参考包含块的网格。
%	参考包含块的高度。
none	
inherit	

Hyperlink 属性

属性	描述	CSS
target	简写属性，设置target-name、target-new以及target-position属性。	3
target-name	规定在何处打开链接（链接的目标）。	3
target-new	规定目标链接在新窗口还是在已有窗口的新标签页中打开。	3
target-position	规定在何处放置新的目标链接。	3

CSS3 target 属性

实例

在新窗口中打开所有超链接，并在所有其他标签页/窗口上面放置新窗口：

```
a
{
target:new front;
}
```

浏览器支持

目前没有浏览器支持 target。

定义和用法

target 属性是一个简写属性，用于设置以下属性：

- target-name
- target-new
- target-position

默认值：	current window above
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.target="new front"

语法

```
target: _target-name_ _target-new_ _target-position_;
```

值	描述
target-name	规定在何处打开超链接（target destination）。
target-new	规定应该在新窗口或已有窗口的新标签页中打开超链接。
target-position	规定在何处放置新的目的地链接。

注释：target-new 和 target-position 值只有在 target-name 值创建新标签页或新窗口中有效。

CSS3 target-name 属性

实例

在新窗口中打开所有超链接：

```
a
{
target-name:new;
}
```

浏览器支持

目前没有浏览器支持 target-name。

定义和用法

target-name 属性规定在何处打开超链接 (target destination)。

默认值：	current
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.targetName="new"

语法

```
target-name: current|root|parent|new|modal|_name_;
```

值	描述
current	在链接所在的框架、标签页或窗口中打开超链接。
root	在当前标签页或窗口中超链接。
parent	在父框架中打开超链接。如果当前框架没有父框架，则将该值视作 root。
new	创建新的目的地（参阅 target-new）。
modal	在新的（暂时创建的）模态窗口中打开新窗口。
name	在已有框架、窗口或标签页中打开链接。如果 name 目的地不存在，则用该名称创建新的目的地。

CSS3 target-new 属性

实例

在新标签页而不是新窗口中打开超链接：

```
a
{
target-name:new;
target-new:tab;
}
```

浏览器支持

目前没有浏览器支持 target-new。

定义和用法

target-new 属性规定在新窗口还是新标签页或已有窗口中打开新的目的地链接。

注释：target-new 属性只有在 target-name 属性创建新标签页或新窗口时有效。

默认值：	window
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.targetNew="tab"

语法

```
target-new: window|tab|none;
```

值	描述
window	在新窗口中打开超链接。
tab	在已有窗口的新标签页中打开超链接。
none	不创建新的目的地。

CSS3 target-position 属性

实例

在新窗口中打开超链接，并在所有其他标签页/窗口之前放置新的窗口：

```
a
{
target-name:new;
target-position:front;
}
```

浏览器支持

目前没有浏览器支持 target-position。

定义和用法

target-position 属性规定在何处放置新的目的地链接。

注释：target-position 属性只有在 target-name 属性创建新标签页或新窗口时有效。

默认值：	above
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.targetPosition="front"

语法

```
target-position: above|behind|front|back;
```

值	描述
above	在当前标签页/窗口之前放置新的目的地标签页/窗口。
behind	在当前标签页/窗口之后放置新的目的地标签页/窗口。
front	在所有其他标签页/窗口之前放置新的目的地标签页/窗口。
back	在所有其他标签页/窗口之后放置新的目的地标签页/窗口。

CSS 列表属性（List）

属性	描述	CSS
list-style	在一个声明中设置所有的列表属性。	1
list-style-image	将图象设置为列表项标记。	1
list-style-position	设置列表项标记的放置位置。	1
list-style-type	设置列表项标记的类型。	1
marker-offset	2	

CSS list-style 属性

实例

把图像设置为列表中的列表项目标记：

```
ul
  {
  list-style:square inside url('/i/arrow.gif');
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 list-style 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

list-style 简写属性在一个声明中设置所有的列表属性。

说明

该属性是一个简写属性，涵盖了所有其他列表样式属性。由于它应用到所有 display 为 list-item 的元素，所以在普通的 HTML 和 XHTML 中只能用于 li 元素，不过实际上它可以应用到任何元素，并由 list-item 元素继承。

可以按顺序设置如下属性：

- list-style-type
- list-style-position
- list-style-image

可以不设置其中的某个值，比如 "list-style:circle inside;" 也是允许的。未设置的属性会使用其默认值。

默认值：	disc outside none
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.listStyle="decimal inside"

可能的值

值	描述
list-style-type	设置列表项标记的类型。参阅：list-style-type 中可能的值。
list-style-position	设置在何处放置列表项标记。参阅：list-style-position 中可能的值。
list-style-image	使用图像来替换列表项的标记。参阅：list-style-image 中可能的值。
inherit	规定应该从父元素继承 list-style 属性的值。

TIY 实例

在一个声明中定义所有的列表属性

本例演示将所有针对列表的属性设置于一个简写属性。

```
<html>
<head>
<style type="text/css">
ul
{
list-style: square inside url('/i/eg_arrow.gif')
}
</style>
</head>

<body>
<ul>
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ul>
</body>

</html>
```

相关页面

CSS 教程：CSS 列表

HTML DOM 参考手册：listStyle 属性

CSS list-style-image 属性

实例

把图像设置为列表中的项目标记：

```
ul
  {
  list-style-image:url("/i/arrow.gif");
  list-style-type:square;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 list-style-image 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

list-style-image 属性使用图像来替换列表项的标记。

说明

这个属性指定作为一个有序或无序列表项标志的图像。图像相对于列表项内容的放置位置通常使用 list-style-position 属性控制。

注释：请始终规定一个 "list-style-type" 属性以防图像不可用。

默认值：	none
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.listStyleImage="url('/images/blueball.gif')"

可能的值

值	描述
URL	图像的路径。
none	默认。无图形被显示。
inherit	规定应该从父元素继承 list-style-image 属性的值。

TIY 实例

将图像作为列表项标记

本例演示如何将图像作为列表项标记。

```
<html>
<head>
<style type="text/css">
ul
{
list-style-image: url('/i/eg_arrow.gif')
}
</style>
</head>

<body>
<ul>
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ul>
</body>

</html>
```

相关页面

CSS 教程：CSS 列表

CSS 参考手册：CSS list-style 属性

HTML DOM 参考手册：listStyleImage 属性

CSS list-style-position 属性

实例

规定列表中列表项目标记的位置：

```
ul
  {
  list-style-position:inside;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 list-style-position 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

list-style-position 属性设置在何处放置列表项标记。

说明

该属性用于声明列表标志相对于列表项内容的位置。外部 (outside) 标志会放在离列表项边框边界一定距离处，不过这距离在 CSS 中未定义。内部 (inside) 标志处理为好像它们是插入在列表项内容最前面的行内元素一样。

默认值：	outside
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.listStylePosition="inside"

可能的值

值	描述
inside	列表项目标记放置在文本以内，且环绕文本根据标记对齐。
outside	默认值。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐。
inherit	规定应该从父元素继承 list-style-position 属性的值。

TIY 实例

放置列表标记

本例演示在何处放置列表标记。

```
<html>
<head>
<style type="text/css">
ul.inside
{
list-style-position: inside
}

ul.outside
{
list-style-position: outside
}
</style>
</head>

<body>
<p>该列表的 list-style-position 的值是 "inside"：</p>
<ul class="inside">
<li>Earl Grey Tea - 一种黑颜色的茶</li>
<li>Jasmine Tea - 一种神奇的“全功能”茶</li>
<li>Honeybush Tea - 一种令人愉快的果味茶</li>
</ul>

<p>该列表的 list-style-position 的值是 "outside"：</p>
<ul class="outside">
<li>Earl Grey Tea - 一种黑颜色的茶</li>
<li>Jasmine Tea - 一种神奇的“全功能”茶</li>
<li>Honeybush Tea - 一种令人愉快的果味茶</li>
</ul>
</body>
</html>
```

相关页面

CSS 教程：CSS 列表

CSS 参考手册：CSS list-style 属性

HTML DOM 参考手册：listStylePosition 属性

CSS list-style-type 属性

实例

设置不同的列表样式：

```
ul.circle {list-style-type:circle;}
ul.square {list-style-type:square;}
ol.upper-roman {list-style-type:upper-roman;}
ol.lower-alpha {list-style-type:lower-alpha;}
```

（在页面底部可以找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 list-style-type 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "decimal-leading-zero"、"lower-greek"、"lower-latin"、"upper-latin"、"armenian"、"georgian" 或 "inherit"。

定义和用法

list-style-type 属性设置列表项标记的类型。

默认值：	**disc**
继承性：	yes
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.listStyleType="square"
---	---

可能的值

CSS2 的值：

值	描述
none	无标记。
disc	默认。标记是实心圆。
circle	标记是空心圆。
square	标记是实心方块。
decimal	标记是数字。
decimal-leading-zero	0开头的数字标记。(01, 02, 03, 等。)
lower-roman	小写罗马数字(i, ii, iii, iv, v, 等。)
upper-roman	大写罗马数字(I, II, III, IV, V, 等。)
lower-alpha	小写英文字母The marker is lower-alpha (a, b, c, d, e, 等。)
upper-alpha	大写英文字母The marker is upper-alpha (A, B, C, D, E, 等。)
lower-greek	小写希腊字母(alpha, beta, gamma, 等。)
lower-latin	小写拉丁字母(a, b, c, d, e, 等。)
upper-latin	大写拉丁字母(A, B, C, D, E, 等。)
hebrew	传统的希伯来编号方式
armenian	传统的亚美尼亚编号方式
georgian	传统的乔治亚编号方式(an, ban, gan, 等。)
cjk-ideographic	简单的表意数字
hiragana	标记是：a, i, u, e, o, ka, ki, 等。（日文片假名）
katakana	标记是：A, I, U, E, O, KA, KI, 等。（日文片假名）
hiragana-iroha	标记是：i, ro, ha, ni, ho, he, to, 等。（日文片假名）
katakana-iroha	标记是：I, RO, HA, NI, HO, HE, TO, 等。（日文片假名）

CSS2.1 的值：

```
disc | circle | square | decimal | decimal-leading-zero |
lower-roman | upper-roman | lower-greek | lower-latin | upper-latin
armenian | georgian | none | inherit
```

TIY 实例

在无序列表中的不同类型的列表标记

本例演示在CSS中不同类型的列表项标记。

```
<html>
<head>
<style type="text/css">
ul.disc {list-style-type: disc}
ul.circle {list-style-type: circle}
ul.square {list-style-type: square}
ul.none {list-style-type: none}
</style>
</head>

<body>
<ul class="disc">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="circle">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="square">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="none">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ul>
</body>

</html>
```

在有序列表中不同类型的列表项标记

本例演示在CSS中不同类型的列表项标记。

```
<html>
<head>
<style type="text/css">
ol.decimal {list-style-type: decimal}
ol.lroman {list-style-type: lower-roman}
ol.uroman {list-style-type: upper-roman}
ol.lalpha {list-style-type: lower-alpha}
ol.ualpha {list-style-type: upper-alpha}
</style>
</head>

<body>
<ol class="decimal">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ol>

<ol class="lroman">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ol>

<ol class="uroman">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ol>

<ol class="lalpha">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ol>

<ol class="ualpha">
<li>咖啡</li>
<li>茶</li>
<li>可口可乐</li>
</ol>
</body>

</html>
```

所有的列表样式类型

本例演示在CSS中所有不同类型的列表项标记。

```
<html>
<head>
<style type="text/css">
ul.none {list-style-type: none}
ul.disc {list-style-type: disc}
ul.circle {list-style-type: circle}
ul.square {list-style-type: square}
ul.decimal {list-style-type: decimal}
ul.decimal-leading-zero {list-style-type: decimal-leading-zero}
ul.lower-roman {list-style-type: lower-roman}
ul.upper-roman {list-style-type: upper-roman}
ul.lower-alpha {list-style-type: lower-alpha}
ul.upper-alpha {list-style-type: upper-alpha}
ul.lower-greek {list-style-type: lower-greek}
ul.lower-latin {list-style-type: lower-latin}
ul.upper-latin {list-style-type: upper-latin}
ul.hebrew {list-style-type: hebrew}
ul.armenian {list-style-type: armenian}
ul.georgian {list-style-type: georgian}
ul.cjk-ideographic {list-style-type: cjk-ideographic}
ul.hiragana {list-style-type: hiragana}
ul.katakana {list-style-type: katakana}
ul.hiragana-iroha {list-style-type: hiragana-iroha}
ul.katakana-iroha {list-style-type: katakana-iroha}
</style>
</head>

<body>
<ul class="none">
<li>"none" 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="disc">
<li>Disc 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="circle">
<li>Circle 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="square">
<li>Square 类型</li>
<li>茶</li>
<li>可口可乐</li>
```

```
</ul>

<ul class="decimal">
<li>Decimal 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="decimal-leading-zero">
<li>Decimal-leading-zero 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="lower-roman">
<li>Lower-roman 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="upper-roman">
<li>Upper-roman 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="lower-alpha">
<li>Lower-alpha 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="upper-alpha">
<li>Upper-alpha 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="lower-greek">
<li>Lower-greek 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="lower-latin">
<li>Lower-latin 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="upper-latin">
<li>Upper-latin 类型</li>
<li>茶</li>
```

```
<li>可口可乐</li>
</ul>

<ul class="hebrew">
<li>Hebrew 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="armenian">
<li>Armenian 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="georgian">
<li>Georgian 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="cjk-ideographic">
<li>Cjk-ideographic 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="hiragana">
<li>Hiragana 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="katakana">
<li>Katakana 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="hiragana-iroha">
<li>Hiragana-iroha 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

<ul class="katakana-iroha">
<li>Katakana-iroha 类型</li>
<li>茶</li>
<li>可口可乐</li>
</ul>

</body>
</html>
```

相关页面

CSS 教程：CSS 列表

CSS 参考手册：CSS list-style 属性

HTML DOM 参考手册：listStyleType 属性

CSS 外边距属性（Margin）

属性	描述	CSS
margin	在一个声明中设置所有外边距属性。	1
margin-bottom	设置元素的下外边距。	1
margin-left	设置元素的左外边距。	1
margin-right	设置元素的右外边距。	1
margin-top	设置元素的上外边距。	1

CSS margin 属性

实例

设置 p 元素的 4 个外边距：

```
p
  {
  margin:2cm 4cm 3cm 4cm;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 margin 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

margin 简写属性在一个声明中设置所有外边距属性。该属性可以有 1 到 4 个值。

说明

这个简写属性设置一个元素所有外边距的宽度，或者设置各边上外边距的宽度。

块级元素的垂直相邻外边距会合并，而行内元素实际上不占上下外边距。行内元素的的左右外边距不会合并。同样地，浮动元素的外边距也不会合并。允许指定负的外边距值，不过使用时要小心。

注释：允许使用负值。

例子 1

```
margin:10px 5px 15px 20px;
```

- 上外边距是 10px
- 右外边距是 5px

- 下外边距是 15px
- 左外边距是 20px

例子 2

```
margin:10px 5px 15px;
```

- 上外边距是 10px
- 右外边距和左外边距是 5px
- 下外边距是 15px

例子 3

```
margin:10px 5px;
```

- 上外边距和下外边距是 10px
- 右外边距和左外边距是 5px

例子 4

```
margin:10px;
```

- 所有 4 个外边距都是 10px

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.margin="10px 5px"

可能的值

值	描述
auto	浏览器计算外边距。
length	规定以具体单位计的外边距值，比如像素、厘米等。默认值是 0px。
%	规定基于父元素的宽度的百分比的外边距。
inherit	规定应该从父元素继承外边距。

TIY 实例

所有的边距属性在一个声明中

本例演示如何将所有的边距属性设置于一个声明中。

```
<html>
<head>
<style type="text/css">
p.margin {margin: 2cm 4cm 3cm 4cm}
</style>
</head>

<body>

<p>这个段落没有指定外边距。</p>

<p class="margin">这个段落带有指定的外边距。这个段落带有指定的外边距。这个段

<p>这个段落没有指定外边距。</p>

</body>

</html>
```

相关页面

CSS 教程：CSS 外边距

HTML DOM 参考手册：margin 属性

CSS margin-bottom 属性

实例

设置 p 元素的下外边距：

```
p
  {
  margin-bottom:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 margin-bottom 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

margin-bottom 属性设置元素的下外边距。

注释：允许使用负值。

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.marginBottom="10px"

可能的值

值	描述
auto	浏览器计算下外边距。
length	规定以具体单位计的下外边距值，比如像素、厘米等。默认值是 0px。
%	规定基于父元素的宽度的百分比的下外边距。
inherit	规定应该从父元素继承下外边距。

TIY 实例

设置文本的下外边距 1

本例演示如何使用厘米值来设置文本的下外边距。

```
<html>
<head>
<style type="text/css">
p.bottommargin {margin-bottom: 2cm}
</style>
</head>

<body>
<p>这个段落没有指定外边距。</p>
<p class="bottommargin">这个段落带有指定的下外边距。</p>
<p>这个段落没有指定外边距。</p>
</body>

</html>
```

设置文本的下外边距 2

本例演示如何使用百分比值来设置文本的下外边距。

```
<html>
<head>
<style type="text/css">
p.bottommargin
{
margin-bottom: 25%
}
</style>
</head>
<body>

<p>This is a paragraph with no margin specified</p>
<p class="bottommargin">This is a paragraph with a specified bottor
<p>This is a paragraph with no margin specified</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 外边距

HTML DOM 参考手册：marginBottom 属性

CSS margin-left 属性

实例

设置 p 元素的左外边距：

```
p
  {
  margin-left:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 margin-left 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

margin-left 属性设置元素的左外边距。

注释：允许使用负值。

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.marginLeft="10px"

可能的值

值	描述
auto	浏览器设置的左外边距。
length	定义固定的左外边距。默认值是0。
%	定义基于父对象总高度的百分比左外边距。
inherit	规定应该从父元素继承左外边距。

TIY 实例

设置文本的左外边距 1

本例演示如何使用厘米值来设置文本的左边距。

```
<html>
<head>
<style type="text/css">
p.leftmargin {margin-left: 2cm}
</style>
</head>

<body>
<p>这个段落没有指定外边距。</p>
<p class="leftmargin">这个段落带有指定的左外边距。</p>
</body>

</html>
```

设置文本的左外边距 2

本例演示如何使用百分比值来设置文本的左边距。

```
<html>
<head>
<style type="text/css">
p.leftmargin
{
margin-left: 25%
}
</style>
</head>
<body>

<p>This is a paragraph with no margin specified</p>
<p class="leftmargin">This is a paragraph with a specified left mar

</body>
</html>
```

相关页面

CSS 教程：CSS 外边距

HTML DOM 参考手册：marginLeft 属性

CSS margin-right 属性

实例

设置 p 元素的右外边距：

```
p
  {
  margin-right:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 margin-right 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

margin-right 属性设置元素的右外边距。

注释：允许使用负值。

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.marginRight="10px"

可能的值

值	描述
auto	浏览器设置的右外边距。
length	定义固定的右外边距。默认值是 0。
%	定义基于父对象总高度的百分比右外边距。
inherit	规定应该从父元素继承右外边距。

TIY 实例

设置文本的右外边距 1

本例演示如何使用厘米值来设置文本的右外边距。

```
<html>
<head>
<style type="text/css">
p.rightmargin {margin-right: 8cm}
</style>
</head>

<body>
<p>这个段落没有指定外边距。</p>
<p class="rightmargin">这个段落带有指定的右外边距。这个段落带有指定的右外边
</body>

</html>
```

设置文本的右外边距 2

本例演示如何使用百分比值来设置文本的右外边距。

```
<html>
<head>
<style type="text/css">
p.rightmargin
{
margin-right:25%
}
</style>
</head>
<body>

<p style="text-align:right">This is a right aligned paragraph with
<p class="rightmargin" style="text-align:right">This is a right al

</body>
</html>
```

相关页面

CSS 教程：CSS 外边距

HTML DOM 参考手册：marginRight 属性

CSS margin-top 属性

实例

设置 p 元素的上外边距：

```
p
  {
  margin-top:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 margin-top 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

margin-top 属性设置元素的上外边距。

注释：允许使用负值。

默认值：	0
继承性：	no
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.marginTop="10px"
---	---

可能的值

值	描述
auto	浏览器设置的上外边距。
length	定义固定的上外边距。默认值是 0。
%	定义基于父对象总高度的百分比上外边距。
inherit	规定应该从父元素继承上外边距。

TIY 实例

设置文本的上外边距 1

本例演示如何使用厘米值来设置文本的顶边距。

```
<html>
<head>
<style type="text/css">
p.topmargin {margin-top: 5cm}
</style>
</head>

<body>
<p>这个段落没有指定外边距。</p>
<p class="topmargin">这个段落带有指定的上外边距。</p>
</body>
</html>
```

设置文本的上外边距 2

本例演示如何使用百分比值来设置文本的顶边距。

```
<html>
<head>
<style type="text/css">
p.topmargin
{
margin-top: 25%
}
</style>
</head>
<body>

<p>This is a paragraph with no margin specified</p>
<p class="topmargin">This is a paragraph with a specified top margi

</body>
</html>
```

相关页面

CSS 教程：CSS 外边距

HTML DOM 参考手册：marginTop 属性

Marquee 属性

属性	描述	CSS
marquee-direction	设置移动内容的方向。	3
marquee-play-count	设置内容移动多少次。	3
marquee-speed	设置内容滚动得多快。	3
marquee-style	设置移动内容的样式。	3

多列属性（Multi-column）

属性	描述	CSS
column-count	规定元素应该被分隔的列数。	3
column-fill	规定如何填充列。	3
column-gap	规定列之间的间隔。	3
column-rule	设置所有 column-rule-* 属性的简写属性。	3
column-rule-color	规定列之间规则的颜色。	3
column-rule-style	规定列之间规则的样式。	3
column-rule-width	规定列之间规则的宽度。	3
column-span	规定元素应该横跨的列数。	3
column-width	规定列的宽度。	3
columns	规定设置 column-width 和 column-count 的简写属性。	3

CSS3 column-count 属性

实例

将 div 元素中的文本分为三列：

```
div
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari 和 Chrome */
column-count:3;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-count 属性。

Firefox 支持替代的 -moz-column-count 属性。

Safari 和 Chrome 支持替代的 -webkit-column-count 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-count 属性。

定义和用法

column-count 属性规定元素应该被划分的列数。

默认值：	auto
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.columnCount=3

语法

```
column-count: _number_|auto;
```

值	描述	测试
number	元素内容将被划分的最佳列数。	测试
auto	由其他属性决定列数，比如 "column-width"。	测试

亲自试一试 - 实例

Column-count

把 div 元素中的文本划分为三列。

```
<!DOCTYPE html>
<html>
<head>
<style>
.newspaper
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;
}
</style>
</head>
<body>

<p><b>注释：</b>Internet Explorer 不支持 column-count 属性。</p>

<div class="newspaper">
人民网北京2月24日电 (记者 刘阳)国家发展改革委近日发出通知，决定自2月25日零时起

此次国内成品油价格调整幅度，是按照现行国内成品油价格形成机制，根据国际市场油价变

通知指出，这次成品油调价后，国家将按照已建立的补贴机制，继续对种粮农民、渔业（含

通知要求，中石油、中石化、中海油三大公司要组织好成品油生产和调运，保持合理库存，
</div>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 多列

CSS3 column-fill 属性

实例

规定如何对列进行填充：

```
div
{
column-fill:auto;
}
```

浏览器支持

主流浏览器都不支持 column-fill 属性。

定义和用法

IE	Firefox	Chrome	Safari	Opera

column-fill 属性规定如何填充列（是否进行协调）。

默认值：	balance
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.columnFill="auto"
---	---

语法

```
column-fill: balance|auto;
```

值	描述
balance	对列进行协调。浏览器应对列长度的差异进行最小化处理。
auto	按顺序对列进行填充，列长度会各有不同。

相关页面

CSS3 教程：CSS3 多列

CSS3 column-gap 属性

实例

规定列间的间隔为 40 像素：

```
div
{
-moz-column-gap:40px; /* Firefox */
-webkit-column-gap:40px; /* Safari 和 Chrome */
column-gap:40px;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-gap 属性。

Firefox 支持替代的 -moz-column-gap 属性。

Safari 和 Chrome 支持替代的 -webkit-column-gap 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-gap 属性。

定义和用法

column-gap 属性规定列之间的间隔。

注释：如果列之间设置了 column-rule，它会在间隔中间显示。

默认值：	**normal**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.columnGap="40px"

语法

```
column-gap: _length_|normal;
```

值	描述	测试
length	把列间的间隔设置为指定的长度。	测试
normal	规定列间间隔为一个常规的间隔。W3C 建议的值是 1em。	测试

亲自试一试 - 实例

Column-gap

将 div 元素中的文本分为三列，并规定列间 30 像素的间隔。

```
<!DOCTYPE html>
<html>
<head>
<style>
.newspaper
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;

-moz-column-gap:30px; /* Firefox */
-webkit-column-gap:30px; /* Safari and Chrome */
column-gap:30px;
}
</style>
</head>
<body>

<p><b>注释：</b>Internet Explorer 不支持 column-count 属性。</p>

<div class="newspaper">
人民网北京2月24日电 (记者 刘阳)国家发展改革委近日发出通知，决定自2月25日零时起

此次国内成品油价格调整幅度，是按照现行国内成品油价格形成机制，根据国际市场油价变

通知指出，这次成品油调价后，国家将按照已建立的补贴机制，继续对种粮农民、渔业（含

通知要求，中石油、中石化、中海油三大公司要组织好成品油生产和调运，保持合理库存，
</div>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 多列

CSS3 column-rule 属性

实例

规定列之间的宽度、样式和颜色规则：

```
div
{
-moz-column-rule:3px outset #ff00ff; /* Firefox */
-webkit-column-rule:3px outset #ff00ff; /* Safari 和 Chrome */
column-rule:3px outset #ff00ff;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-rule 属性。

Firefox 支持替代的 -moz-column-rule 属性。

Safari 和 Chrome 支持替代的 -webkit-column-rule 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-rule 属性。

定义和用法

column-rule 属性是一个简写属性，用于设置所有 column-rule-* 属性。

column-rule 属性设置列只觉得宽度、样式和颜色规则。

默认值：	medium none black
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.columnRule="3px outset #ff00ff"

语法

```
column-rule: _column-rule-width_ _column-rule-style_ _column-rule-c
```

值	描述
column-rule-width	设置列之间的宽度规则。
column-rule-style	设置列之间的样式规则。
column-rule-color	设置列之间的颜色规则。

亲自试一试 - 实例

Column-rule

规定列之间的宽度、样式和颜色。

```
<!DOCTYPE html>
<html>
<head>
<style>
.newspaper
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;

-moz-column-gap:40px; /* Firefox */
-webkit-column-gap:40px; /* Safari and Chrome */
column-gap:40px;

-moz-column-rule:4px outset #ff0000; /* Firefox */
-webkit-column-rule:4px outset #ff0000; /* Safari and Chrome */
column-rule:4px outset #ff0000;
}
</style>
</head>
<body>

<p><b>注释：</b>Internet Explorer 不支持 column-count 属性。</p>

<div class="newspaper">
人民网北京2月24日电 (记者 刘阳)国家发展改革委近日发出通知，决定自2月25日零时起

此次国内成品油价格调整幅度，是按照现行国内成品油价格形成机制，根据国际市场油价变

通知指出，这次成品油调价后，国家将按照已建立的补贴机制，继续对种粮农民、渔业（含

通知要求，中石油、中石化、中海油三大公司要组织好成品油生产和调运，保持合理库存，
</div>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 多列

CSS3 column-rule-color 属性

实例

设置列之间的颜色规则：

```
div
{
-moz-column-rule-color:#ff0000; /* Firefox */
-webkit-column-rule-color:#ff0000; /* Safari 和 Chrome */
column-rule-color:#ff0000;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-rule-color 属性。

Firefox 支持替代的 -moz-column-rule-color 属性。

Safari 和 Chrome 支持替代的 -webkit-column-rule-color 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-rule-color 属性。

定义和用法

column-rule-color 属性规定列之间的颜色规则。

默认值：	black
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.columnRuleColor="#ff00ff"
---	---

语法

```
column-rule-color: _color_;
```

值	描述
color	规定颜色规则。请参阅 CSS 颜色值。

相关页面

CSS3 教程：CSS3 多列

CSS3 column-rule-style 属性

实例

设置列之间的颜色规则：

```
div
{
-moz-column-rule-style:dotted; /* Firefox */
-webkit-column-rule-style:dotted; /* Safari 和 Chrome */
column-rule-style:dotted;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-rule-style 属性。

Firefox 支持替代的 -moz-column-rule-style 属性。

Safari 和 Chrome 支持替代的 -webkit-column-rule-style 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-rule-style 属性。

定义和用法

column-rule-style 属性规定列之间的样式规则。

默认值：	none
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.columnRuleStyle="dotted"
---	---

语法

```
column-rule-style: none|hidden|dotted|dashed|solid|double|groove|ri
```

值	描述	测试
none	定义没有规则。	测试
hidden	定义隐藏规则。	测试
dotted	定义点状规则。	测试
dashed	定义虚线规则。	测试
solid	定义实线规则。	测试
double	定义双线规则。	测试
groove	定义 3D grooved 规则。该效果取决于宽度和颜色值。	测试
ridge	定义 3D ridged 规则。该效果取决于宽度和颜色值。	测试
inset	定义 3D inset 规则。该效果取决于宽度和颜色值。	测试
outset	定义 3D outset 规则。该效果取决于宽度和颜色值。	测试

相关页面

CSS3 教程：CSS3 多列

CSS3 column-rule-width 属性

实例

设置列之间的颜色规则：

```
div
{
-moz-column-rule-width:1px; /* Firefox */
-webkit-column-rule-width:1px; /* Safari 和 Chrome */
column-rule-width:1px;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-rule-width 属性。

Firefox 支持替代的 -moz-column-rule-width 属性。

Safari 和 Chrome 支持替代的 -webkit-column-rule-width 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-rule-width 属性。

定义和用法

column-rule-width 属性规定列之间的宽度规则。

默认值：	medium
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.columnRuleWidth="thin"

语法

```
column-rule-width: thin|medium|thick|_length_;
```

值	描述	测试
thin	定义纤细规则。	测试
medium	定义中等规则。	测试
thick	定义宽厚规则。	测试
length	规定规则的宽度。	测试

相关页面

CSS3 教程：CSS3 多列

CSS3 column-span 属性

实例

使 h2 元素横跨所有列：

```
h2
{
-webkit-column-span:all; /* Chrome */
column-span:all;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-span 属性。

Safari 和 Chrome 支持替代的 -webkit-column-span 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-span 属性。

定义和用法

column-span 属性规定元素应横跨多少列。

默认值：	1
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.columnSpan="all"
---	---

语法

```
column-span: 1|all;
```

值	描述	测试
1	元素应横跨一列。	测试
all	元素应横跨所有列。	测试

相关页面

CSS3 教程：CSS3 多列

CSS3 column-width 属性

实例

规定列的宽度：

```
div
{
column-width:100px;
-moz-column-width:100px; /* Firefox */
-webkit-column-width:100px; /* Safari 和 Chrome */
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column-width 属性。

Firefox 支持替代的 -moz-column-width 属性。

Safari 和 Chrome 支持替代的 -webkit-column-width 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column-width 属性。

定义和用法

column-width 属性规定列的宽度。

默认值：	auto
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.columnWidth="100px"

语法

```
column-width: auto|_length_;
```

值	描述	测试
auto	由浏览器决定列宽。	测试
length	规定列的宽度。	测试

相关页面

CSS3 教程：CSS3 多列

CSS3 columns 属性

实例

规定列的宽度和列数：

```
div
{
columns:100px 3;
-moz-columns:100px 3; /* Firefox */
-webkit-columns:100px 3; /* Safari 和 Chrome */
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10 和 Opera 支持 column 属性。

Firefox 支持替代的 -moz-column 属性。

Safari 和 Chrome 支持替代的 -webkit-column 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 column 属性。

定义和用法

columns 属性是一个简写属性，用于设置列宽和列数。

默认值：	auto auto
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.columns="100px 3"

语法

```
columns: _column-width_ _column-count_;
```

值	描述
column-width	列的宽度。
column-count	列数。

相关页面

CSS3 教程：CSS3 多列

CSS 内边距属性（Padding）

属性	描述	CSS
padding	在一个声明中设置所有内边距属性。	1
padding-bottom	设置元素的下内边距。	1
padding-left	设置元素的左内边距。	1
padding-right	设置元素的右内边距。	1
padding-top	设置元素的上内边距。	1

CSS padding 属性

实例

设置 p 元素的 4 个内边距：

```
p
  {
  padding:2cm 4cm 3cm 4cm;
  }
```

浏览器支持

所有浏览器都支持 padding 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

padding 简写属性在一个声明中设置所有内边距属性。

说明

这个简写属性设置元素所有内边距的宽度，或者设置各边上内边距的宽度。行内非替换元素上设置的内边距不会影响行高计算；因此，如果一个元素既有内边距又有背景，从视觉上看可能会延伸到其他行，有可能还会与其他内容重叠。元素的背景会延伸穿过内边距。不允许指定负边距值。

注释：不允许使用负值。

例子 1

```
padding:10px 5px 15px 20px;
```

- 上内边距是 10px
- 右内边距是 5px
- 下内边距是 15px
- 左内边距是 20px

例子 2

```
padding:10px 5px 15px;
```

- 上内边距是 10px
- 右内边距和左内边距是 5px
- 下内边距是 15px

例子 3

```
padding:10px 5px;
```

- 上内边距和下内边距是 10px
- 右内边距和左内边距是 5px

例子 4

```
padding:10px;
```

- 所有 4 个内边距都是 10px

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.padding="10px 5px"

可能的值

值	描述
auto	浏览器计算内边距。
length	规定以具体单位计的内边距值，比如像素、厘米等。默认值是 0px。
%	规定基于父元素的宽度的百分比的内边距。
inherit	规定应该从父元素继承内边距。

TIY 实例

所有内边距属性在一个声明中

本例演示使用简写属性将所有的内边距属性设置于一个声明中，可以有一到四个值。

```
<html>
<head>
<style type="text/css">
td.test1 {padding: 1.5cm}
td.test2 {padding: 0.5cm 2.5cm}
</style>
</head>

<body>
<table border="1">
<tr>
<td class="test1">
这个表格单元的每个边拥有相等的内边距。
</td>
</tr>
</table>
<br />
<table border="1">
<tr>
<td class="test2">
这个表格单元的上和下内边距是 0.5cm，左和右内边距是 2.5cm。
</td>
</tr>
</table>
</body>

</html>
```

相关页面

CSS 教程：CSS 内边距

HTML DOM 参考手册：padding 属性

CSS padding-bottom 属性

实例

设置 p 元素的下内边距：

```
p
  {
  padding-bottom:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

所有主流浏览器都支持 padding-bottom 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

padding-bottom 属性设置元素的下内边距（底部空白）。

说明

该属性设置元素下内边距的宽度。行内非替换元素上设置的下内边距不会影响行高计算，因此，如果一个元素既有内边距又有背景，从视觉上看可能延伸到其他行，有可能还会与其他内容重叠。不允许指定负内边距值。

注释：不允许使用负值。

默认值：	0
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.paddingBottom="10px"

可能的值

值	描述
length	规定以具体单位计的固定的下内边距值，比如像素、厘米等。默认值是 0px。
%	定义基于父元素宽度的百分比下内边距。此值不会如预期地那样工作于所有的浏览器中。
inherit	规定应该从父元素继承下内边距。

TIY 实例

设置下内边距 1

本例演示如何使用厘米值来设置单元格的下内边距。

```
<html>
<head>
<style type="text/css">
td {padding-bottom: 2cm}
</style>
</head>

<body>
<table border="1">
<tr>
<td>
这个表格单元拥有下内边距。
</td>
</tr>
</table>
</body>

</html>
```

设置下内边距 2

本例演示如何使用百分比值来设置单元格的下内边距。

```
<html>
<head>
<style type="text/css">
td
{
padding-bottom: 10%
}
</style>
</head>
<body>

<table border="1">
<tr>
<td>
这个表格单元拥有下内边距。
</td>
</tr>
</table>

</body>
</html>
```

相关页面

CSS 教程：CSS 内边距

HTML DOM 参考手册：paddingBottom 属性

CSS padding-left 属性

实例

设置 p 元素的左内边距：

```
p
  {
  padding-left:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

所有主流浏览器都支持 padding-left 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

padding-left 属性设置元素左内边距（空白）。

说明

该属性设置元素左内边距的宽度。行内非替换元素上设置的左内边距仅在元素所生成的第一个行内框的左边出现.

注释：不允许使用负值。

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.paddingLeft="10px"

可能的值

值	描述
length	规定以具体单位计的固定的左内边距值，比如像素、厘米等。默认值是 0px。
%	定义基于父元素宽度的百分比左内边距。此值不会如预期地那样工作于所有的浏览器中。
inherit	规定应该从父元素继承左内边距。

TIY 实例

设置左内边距 1

本例演示如何使用厘米值来设置单元格的左内边距。

```
<html>
<head>
<style type="text/css">
td {padding-left: 2cm}
</style>
</head>

<body>
<table border="1">
<tr>
<td>
这个表格单元拥有左内边距。
</td>
</tr>
</table>
</body>

</html>
```

设置左内边距 2

本例演示如何使用百分比值来设置单元格的左内边距。

```
<html>
<head>
<style type="text/css">
td
{
padding-left: 10%
}
</style>
</head>
<body>

<table border="1">
<tr>
<td>
这个表格单元拥有左内边距。
</td>
</tr>
</table>

</body>
</html>
```

相关页面

CSS 教程：CSS 内边距

HTML DOM 参考手册：paddingLeft 属性

CSS padding-right 属性

实例

设置 p 元素的右内边距：

```
p
  {
  padding-right:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

所有主流浏览器都支持 padding-right 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

padding-right 属性设置元素右内边距（空白）。

注释：不允许使用负值。

说明

该属性设置元素右内边距的宽度。行内非替换元素上设置的右内边距仅在元素所生成的第一个行内框的右边出现.

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.paddingRight="10px"

可能的值

值	描述
length	规定以具体单位计的固定的右内边距值，比如像素、厘米等。默认值是 0px。
%	定义基于父元素宽度的百分比右内边距。此值不会如预期地那样工作于所有的浏览器中。
inherit	规定应该从父元素继承右内边距。

TIY 实例

设置右内边距 1

本例演示如何使用厘米值来设置单元格的右内边距。

```
<html>
<head>
<style type="text/css">
td {padding-right: 5cm}
</style>
</head>

<body>
<table border="1">
<tr>
<td>
这个表格单元拥有右内边距。
</td>
</tr>
</table>
</body>

</html>
```

设置右内边距 2

本例演示如何使用百分比值来设置单元格的右内边距。

```
<html>
<head>
<style type="text/css">
td
{
padding-right: 10%
}
</style>
</head>
<body>

<table border="1">
<tr>
<td>
这个表格单元拥有右内边距。
</td>
</tr>
</table>

</body>
</html>
```

相关页面

CSS 教程：CSS 内边距

HTML DOM 参考手册：paddingRight 属性

CSS padding-top 属性

实例

设置 p 元素的上内边距：

```
p
  {
  padding-top:2cm;
  }
```

（在页面底部可以找到更多实例）

浏览器支持

所有主流浏览器都支持 padding-top 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

padding-top 属性设置元素的上内边距（空间）。

说明

该属性设置元素上内边距的宽度。行内非替换元素上设置的上内边距不会影响行高计算，因此，如果一个元素既有内边距又有背景，从视觉上看可能延伸到其他行，有可能还会与其他内容重叠。不允许指定负内边距值。

注释：不允许使用负值。

默认值：	**0**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.paddingTop="10px"

可能的值

值	描述
length	规定以具体单位计的固定的上内边距值，比如像素、厘米等。默认值是 0px。
%	定义基于父元素宽度的百分比上内边距。此值不会如预期的那样工作于所有的浏览器中。
inherit	规定应该从父元素继承上内边距。

TIY 实例

设置上内边距 1

本例演示如何使用厘米值来设置单元格的上内边距。

```
<html>
<head>
<style type="text/css">
td {padding-top: 2cm}
</style>
</head>

<body>
<table border="1">
<tr>
<td>
这个表格单元拥有上内边距。
</td>
</tr>
</table>
</body>

</html>
```

设置上内边距 2

本例演示如何使用百分比值来设置单元格的上内边距。

```
<html>
<head>
<style type="text/css">
td
{
padding-top: 10%
}
</style>
</head>
<body>

<table border="1">
<tr>
<td>
这个表格单元拥有上内边距。
</td>
</tr>
</table>

</body>
</html>
```

相关页面

CSS 教程：CSS 内边距

HTML DOM 参考手册：paddingTop 属性

Paged Media 属性

属性	描述	CSS
fit	示意如何对width和height属性均不是auto的被替换元素进行缩放。	3
fit-position	定义盒内对象的对齐方式。	3
image-orientation	规定用户代理应用于图像的顺时针方向旋转。	3
page	规定元素应该被显示的页面特定类型。	3
size	规定页面内容包含框的尺寸和方向。	3

CSS 定位属性（Positioning）

属性	描述	CSS
bottom	设置定位元素下外边距边界与其包含块下边界之间的偏移。	2
clear	规定元素的哪一侧不允许其他浮动元素。	1
clip	剪裁绝对定位元素。	2
cursor	规定要显示的光标的类型（形状）。	2
display	规定元素应该生成的框的类型。	1
float	规定框是否应该浮动。	1
left	设置定位元素左外边距边界与其包含块左边界之间的偏移。	2
overflow	规定当内容溢出元素框时发生的事情。	2
position	规定元素的定位类型。	2
right	设置定位元素右外边距边界与其包含块右边界之间的偏移。	2
top	设置定位元素的上外边距边界与其包含块上边界之间的偏移。	2
vertical-align	设置元素的垂直对齐方式。	1
visibility	规定元素是否可见。	2
z-index	设置元素的堆叠顺序。	2

CSS bottom 属性

实例

把图像的底边缘设置在其包含元素底边缘之上 5 像素高的位置：

```
img
  {
  position:absolute;
  bottom:5px;
  }
```

浏览器支持

所有主流浏览器都支持 bottom 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

bottom 属性规定元素的底部边缘。该属性定义了定位元素下外边距边界与其包含块下边界之间的偏移。

注释：如果 "position" 属性的值为 "static"，那么设置 "bottom" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，如果 bottom 和 top 都是 auto，其计算值则都是 0；如果其中之一为 auto，则取另一个值的相反数；如果二者都不是 auto，bottom 将取 top 值的相反数。

默认值：	auto
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.bottom="50px"

可能的值

值	描述
auto	默认值。通过浏览器计算底部的位置。
%	设置以包含元素的百分比计的底边位置。可使用负值。
length	使用 px、cm 等单位设置元素的底边位置。可使用负值。
inherit	规定应该从父元素继承 bottom 属性的值。

TIY 实例

使用像素值设置图像的底部边缘

本例演示如何使用像素值设置图像的底部边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
bottom:0px
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

使用百分比设置图像的底部边缘

本例演示如何使用百分比值设置图像的底部边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
bottom:5%
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：bottom 属性

CSS clear 属性

实例

图像的左侧和右侧均不允许出现浮动元素：

```
img
  {
  float:left;
  clear:both;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 clear 属性。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

clear 属性规定元素的哪一侧不允许其他浮动元素。

说明

clear 属性定义了元素的哪边上不允许出现浮动元素。在 CSS1 和 CSS2 中，这是通过自动为清除元素（即设置了 clear 属性的元素）增加上外边距实现的。在 CSS2.1 中，会在元素上外边距之上增加清除空间，而外边距本身并不改变。不论哪一种改变，最终结果都一样，如果声明为左边或右边清除，会使元素的上外边框边界刚好在该边上浮动元素的下外边距边界之下。

默认值：	**none**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.clear="left"

可能的值

值	描述
left	在左侧不允许浮动元素。
right	在右侧不允许浮动元素。
both	在左右两侧均不允许浮动元素。
none	默认值。允许浮动元素出现在两侧。
inherit	规定应该从父元素继承 clear 属性的值。

TIY 实例

清除元素的侧面

本例演示如何使用清除元素侧面的浮动元素。

```
<html>

<head>
<style type="text/css">
img
  {
  float:left;
  clear:both;
  }
</style>
</head>

<body>
<img src="/i/eg_smile.gif" />
<img src="/i/eg_smile.gif" />
</body>

</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：clear 属性

CSS clip 属性

实例

剪裁图像：

```
img
  {
  position:absolute;
  clip:rect(0px,60px,200px,0px);
  }
```

浏览器支持

所有主流浏览器都支持 clip 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

clip 属性剪裁绝对定位元素。

当一幅图像的尺寸大于包含它的元素时会发生什么呢？"clip" 属性允许您规定一个元素的可见尺寸，这样此元素就会被修剪并显示为这个形状。

说明

这个属性用于定义一个剪裁矩形。对于一个绝对定义元素，在这个矩形内的内容才可见。出了这个剪裁区域的内容会根据 overflow 的值来处理。剪裁区域可能比元素的内容区大，也可能比内容区小。

默认值：	**auto**
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.clip="rect(0px,50px,50px,0px)"
---	---

可能的值

值	描述
shape	设置元素的形状。唯一合法的形状值是：rect (*top*, *right*, *bottom*, *left*)
auto	默认值。不应用任何剪裁。
inherit	规定应该从父元素继承 clip 属性的值。

TIY 实例

设置元素的形状

本例演示如何设置元素的形状。此元素被剪入形状中，然后显示出来。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
clip:rect(0px 50px 200px 0px)
}
</style>
</head>

<body>
<p>clip 属性剪切了一幅图像：</p>
<p><img border="0" src="/i/eg_bookasp.gif" width="120" height="151'
</body>

</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：clip 属性

CSS cursor 属性

实例

一些不同的光标：

```
span.crosshair {cursor:crosshair;}
span.help {cursor:help;}
span.wait {cursor:wait;}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 cursor 属性。

注释：Opera 9.3 和 Safari 3 不支持 *url* 值。

注释：任何版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

cursor 属性规定要显示的光标的类型（形状）。

该属性定义了鼠标指针放在一个元素边界范围内时所用的光标形状（不过 CSS2.1 没有定义由哪个边界确定这个范围）。

默认值：	auto
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.cursor="crosshair"

可能的值

值	描述
url	

需使用的自定义光标的 URL。

注释：请在此列表的末端始终定义一种普通的光标，以防没有由 URL 定义的可用光标。

| | default | 默认光标（通常是一个箭头） | | auto | 默认。浏览器设置的光标。 | | crosshair | 光标呈现为十字线。 | | pointer | 光标呈现为指示链接的指针（一只手） | | move | 此光标指示某对象可被移动。 | | e-resize | 此光标指示矩形框的边缘可被向右（东）移动。 | | ne-resize | 此光标指示矩形框的边缘可被向上及向右移动（北/东）。 | | nw-resize | 此光标指示矩形框的边缘可被向上及向左移动（北/西）。 | | n-resize | 此光标指示矩形框的边缘可被向上（北）移动。 | | se-resize | 此光标指示矩形框的边缘可被向下及向右移动（南/东）。 | | sw-resize | 此光标指示矩形框的边缘可被向下及向左移动（南/西）。 | | s-resize | 此光标指示矩形框的边缘可被向下移动（南）。 | | w-resize | 此光标指示矩形框的边缘可被向左移动（西）。 | | text | 此光标指示文本。 | | wait | 此光标指示程序正忙（通常是一只表或沙漏）。 | | help | 此光标指示可用的帮助（通常是一个问号或一个气球）。 |

TIY 实例

改变光标

本例演示如何改变光标。

```
<html>

<body>
<p>请把鼠标移动到单词上，可以看到鼠标指针发生变化：</p>
<span style="cursor:auto">
Auto</span><br />
<span style="cursor:crosshair">
Crosshair</span><br />
<span style="cursor:default">
Default</span><br />
<span style="cursor:pointer">
Pointer</span><br />
<span style="cursor:move">
Move</span><br />
<span style="cursor:e-resize">
e-resize</span><br />
<span style="cursor:ne-resize">
ne-resize</span><br />
<span style="cursor:nw-resize">
nw-resize</span><br />
<span style="cursor:n-resize">
n-resize</span><br />
<span style="cursor:se-resize">
se-resize</span><br />
<span style="cursor:sw-resize">
sw-resize</span><br />
<span style="cursor:s-resize">
s-resize</span><br />
<span style="cursor:w-resize">
w-resize</span><br />
<span style="cursor:text">
text</span><br />
<span style="cursor:wait">
wait</span><br />
<span style="cursor:help">
help</span>
</body>

</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：cursor 属性

CSS display 属性

实例

使段落生出行内框：

```
p.inline
  {
  display:inline;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 display 属性。

注释：如果规定了 !DOCTYPE，则 Internet Explorer 8 （以及更高版本）支持属性值 "inline-table"、"run-in"、"table"、"table-caption"、"table-cell"、"table-column"、"table-column-group"、"table-row"、"table-row-group"、以及 "inherit"。

定义和用法

display 属性规定元素应该生成的框的类型。

说明

这个属性用于定义建立布局时元素生成的显示框类型。对于 HTML 等文档类型，如果使用 display 不谨慎会很危险，因为可能违反 HTML 中已经定义的显示层次结构。对于 XML，由于 XML 没有内置的这种层次结构，所有 display 是绝对必要的。

注释：CSS2 中有值 compact 和 marker，不过由于缺乏广泛的支持，已经从 CSS2.1 中去除了。

默认值：	**inline**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.display="inline"

可能的值

值	描述
none	此元素不会被显示。
block	此元素将显示为块级元素，此元素前后会带有换行符。
inline	默认。此元素会被显示为内联元素，元素前后没有换行符。
inline-block	行内块元素。（CSS2.1 新增的值）
list-item	此元素会作为列表显示。
run-in	此元素会根据上下文作为块级元素或内联元素显示。
compact	CSS 中有值 compact，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。
marker	CSS 中有值 marker，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。
table	此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。
inline-table	此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。
table-row-group	此元素会作为一个或多个行的分组来显示（类似 <tbody>）。
table-header-group	此元素会作为一个或多个行的分组来显示（类似 <thead>）。
table-footer-group	此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。
table-row	此元素会作为一个表格行显示（类似 <tr>）。
table-column-group	此元素会作为一个或多个列的分组来显示（类似 <colgroup>）。
table-column	此元素会作为一个单元格列显示（类似 <col>）
table-cell	此元素会作为一个表格单元格显示（类似 <td> 和 <th>）
table-caption	此元素会作为一个表格标题显示（类似 <caption>）
inherit	规定应该从父元素继承 display 属性的值。

TIY 实例

如何把元素显示为内联元素

本例演示如何把元素显示为内联元素。

```
<html>
<head>
<style type="text/css">
p {display: inline}
div {display: none}
</style>
</head>

<body>
<p>本例中的样式表把段落元素设置为内联元素。</p>

<p>而 div 元素不会显示出来！</p>

<div>div 元素的内容不会显示出来！</div>
</body>
</html>
```

如何把元素显示为块级元素

本例演示如何把元素显示为块级元素。

```
<html>
<head>
<style type="text/css">
span
{
display: block
}
</style>
</head>
<body>

<span>本例中的样式表把 span 元素设置为块级元素。</span>
<span>两个 span 元素之间产生了一个换行行为。</span>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：display 属性

CSS float 属性

实例

把图像向右浮动：

```
img
  {
  float:right;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 float 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

float 属性定义元素在哪个方向浮动。以往这个属性总应用于图像，使文本围绕在图像周围，不过在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。

如果浮动非替换元素，则要指定一个明确的宽度；否则，它们会尽可能地窄。

注释：假如在一行之上只有极少的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。

默认值：	none
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.cssFloat="left"

可能的值

值	描述
left	元素向左浮动。
right	元素向右浮动。
none	默认值。元素不浮动，并会显示在其在文本中出现的位置。
inherit	规定应该从父元素继承 float 属性的值。

TIY 实例

float 属性的简单应用

使图像浮动于一个段落的右侧。

```
<html>
<head>
<style type="text/css">
img
{
float:right
}
</style>
</head>

<body>
<p>在下面的段落中，我们添加了一个样式为 <b>float:right</b> 的图像。结果是这
<p>
<img src="/i/eg_cute.gif" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>

</html>
```

将带有边框和边界的图像浮动于段落的右侧

使图像浮动于段落的右侧。向图像添加边框和边界。

```
<html>
<head>
<style type="text/css">
img
{
float:right;
border:1px dotted black;
margin:0px 0px 15px 20px;
}
</style>
</head>

<body>
<p>在下面的段落中，图像会浮动到右侧，并且添加了点状的边框。我们还为图像添加了边
<p>
<img src="/i/eg_cute.gif" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>

</html>
```

带标题的图像浮动于右侧

使带有标题的图像浮动于右侧

```
<html>
<head>
<style type="text/css">
div
{
float:right;
width:120px;
margin:0 0 15px 20px;
padding:15px;
border:1px solid black;
text-align:center;
}
</style>
</head>

<body>
<div>
<img src="/i/eg_cute.gif" /><br />
CSS is fun!
</div>
<p>
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>

<p>
在上面的段落中，div 元素的宽度是 120 像素，它其中包含图像。div 元素浮动到右侧
</p>
</body>

</html>
```

使段落的首字母浮动于左侧

使段落的首字母浮动于左侧，并向这个字母添加样式。

```
<html>
<head>
<style type="text/css">
span
{
float:left;
width:0.7em;
font-size:400%;
font-family:algerian,courier;
line-height:80%;
}
</style>
</head>

<body>
<p>
<span>T</span>his is some text.
This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>

<p>
在上面的段落中，文本的第一个字母包含在一个 span 元素中。这个 span 元素的宽度是
</p>

</body>
</html>
```

创建水平菜单

使用具有一栏超链接的浮动来创建水平菜单。

```
<html>
<head>
<style type="text/css">
ul
{
float:left;
width:100%;
padding:0;
margin:0;
list-style-type:none;
}
a
{
float:left;
width:7em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300}
li {display:inline}
</style>
</head>

<body>
<ul>
<li><a href="#">Link one</a></li>
<li><a href="#">Link two</a></li>
<li><a href="#">Link three</a></li>
<li><a href="#">Link four</a></li>
</ul>

<p>
在上面的例子中，我们把 ul 元素和 a 元素浮向左浮动。li 元素显示为行内元素（元素
</p>

</body>
</html>
```

创建无表格的首页

使用浮动来创建拥有页眉、页脚、左侧目录和主体内容的首页。

```
<html>
<head>
<style type="text/css">
div.container
{
```

```
width:100%;
margin:0px;
border:1px solid gray;
line-height:150%;
}
div.header,div.footer
{
padding:0.5em;
color:white;
background-color:gray;
clear:left;
}
h1.header
{
padding:0;
margin:0;
}
div.left
{
float:left;
width:160px;
margin:0;
padding:1em;
}
div.content
{
margin-left:190px;
border-left:1px solid gray;
padding:1em;
}
</style>
</head>
<body>

<div class="container">

<div class="header"><h1 class="header">W3School.com.cn</h1></div>

<div class="left"><p>"Never increase, beyond what is necessary, the

<div class="content">
<h2>Free Web Building Tutorials</h2>
<p>At W3School.com.cn you will find all the Web-building tutorials
from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.
<p>W3School.com.cn - The Largest Web Developers Site On The Net!</p

<div class="footer">Copyright 2008 by YingKe Investment.</div>
</div>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：cssFloat 属性

CSS left 属性

实例

把图像的左边缘设置在其包含元素左边缘向右 100 像素的位置：

```
img
  {
  position:absolute;
  left:100px;
  }
```

浏览器支持

所有主流浏览器都支持 left 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

left 属性规定元素的左边缘。该属性定义了定位元素左外边距边界与其包含块左边界之间的偏移。

注释：如果 "position" 属性的值为 "static"，那么设置 "left" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，left 的计算值始终等于 right。

默认值：	auto
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.left="100px"
---	---

可能的值

值	描述
auto	默认值。通过浏览器计算左边缘的位置。
%	设置以包含元素的百分比计的左边位置。可使用负值。
length	使用 px、cm 等单位设置元素的左边位置。可使用负值。
inherit	规定应该从父元素继承 left 属性的值。

TIY 实例

使用固定值设置图像的左边缘

本例演示如何使用固定值设置图像的左边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
left:100px
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

使用百分比设置图像的左边缘

本例演示如何使用百分比值设置图像的左边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
left:20%
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：left 属性

CSS overflow 属性

实例

设置 overflow 属性：

```
div
  {
  width:150px;
  height:150px;
  overflow:scroll;
  }
```

浏览器支持

所有主流浏览器都支持 overflow 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

overflow 属性规定当内容溢出元素框时发生的事情。

说明

这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户代理都会提供一种滚动机制。因此，有可能即使元素框中可以放下所有内容也会出现滚动条。

默认值：	**visible**
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.overflow="scroll"

可能的值

值	描述
visible	默认值。内容不会被修剪，会呈现在元素框之外。
hidden	内容会被修剪，并且其余内容是不可见的。
scroll	内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。
auto	如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。
inherit	规定应该从父元素继承 overflow 属性的值。

TIY 实例

如何使用滚动条来显示元素内溢出的内容

本例演示当元素内容太大而超出规定区域时，如何设置溢出属性来规定相应的动作。

```
<html>
<head>
<style type="text/css">
div
{
background-color:#00FFFF;
width:150px;
height:150px;
overflow: scroll
}
</style>
</head>

<body>
<p>如果元素中的内容超出了给定的宽度和高度属性，overflow 属性可以确定是否显示滚

<div>
这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户
</div>
</body>

</html>
```

如何隐藏溢出元素中溢出的内容

本例演示在元素中的内容太大以至于无法适应指定的区域时，如何设置 overflow 属性来隐藏其内容。

```
<html>
<head>
<style type="text/css">
div
{
background-color:#00FFFF;
width:150px;
height:150px;
overflow: hidden
}
</style>
</head>

<body>
<p>如果元素中的内容超出了给定的宽度和高度属性，overflow 属性可以确定是否显示滚

<div>
这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户
</div>
</body>

</html>
```

如何设置浏览器来自动地处理溢出

本例演示如何设置浏览器来自动地处理溢出。

```
<html>
<head>
<style type="text/css">
div
{
background-color:#00FFFF;
width:150px;
height:150px;
overflow: auto
}
</style>
</head>

<body>
<p>如果元素中的内容超出了给定的宽度和高度属性，overflow 属性可以确定是否显示滚

<div>
这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户
</div>
</body>

</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：overflow 属性

CSS position 属性

实例

定位 h2 元素：

```
h2
  {
  position:absolute;
  left:100px;
  top:150px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 position 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

position 属性规定元素的定位类型。

说明

这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在正常流中的默认位置偏移。

默认值：	static
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.position="absolute"

可能的值

值	描述
absolute	

生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。

元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

| | fixed |

生成绝对定位的元素，相对于浏览器窗口进行定位。

元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

| | relative |

生成相对定位的元素，相对于其正常位置进行定位。

因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。

| | static | 默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。 | | inherit | 规定应该从父元素继承 position 属性的值。 |

TIY 实例

定位：相对定位

本例演示如何相对于一个元素的正常位置来对其定位。

```
<html>
<head>
<style type="text/css">
h2.pos_left
{
position:relative;
left:-20px
}
h2.pos_right
{
position:relative;
left:20px
}
</style>
</head>

<body>
<h2>这是位于正常位置的标题</h2>
<h2 class="pos_left">这个标题相对于其正常位置向左移动</h2>
<h2 class="pos_right">这个标题相对于其正常位置向右移动</h2>
<p>相对定位会按照元素的原始位置对该元素进行移动。</p>
<p>样式 "left:-20px" 从元素的原始左侧位置减去 20 像素。</p>
<p>样式 "left:20px" 向元素的原始左侧位置增加 20 像素。</p>
</body>

</html>
```

定位：绝对定位

本例演示如何使用绝对值来对元素进行定位。

```
<html>
<head>
<style type="text/css">
h2.pos_abs
{
position:absolute;
left:100px;
top:150px
}
</style>
</head>

<body>
<h2 class="pos_abs">这是带有绝对定位的标题</h2>
<p>通过绝对定位，元素可以放置到页面上的任何位置。下面的标题距离页面左侧 100px,
</body>

</html>
```

定位：固定定位

本例演示如何相对于浏览器窗口来对元素进行定位。

```
<html>
<head>
<style type="text/css">
p.one
{
position:fixed;
left:5px;
top:5px;
}
p.two
{
position:fixed;
top:30px;
right:5px;
}
</style>
</head>
<body>

<p class="one">一些文本。</p>
<p class="two">更多的文本。</p>

</body>
</html>
```

设置元素的形状

本例演示如何设置元素的形状。此元素被剪裁到这个形状内，并显示出来。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
clip:rect(0px 50px 200px 0px)
}
</style>
</head>

<body>
<p>clip 属性剪切了一幅图像：</p>
<p><img border="0" src="/i/eg_bookasp.gif" width="120" height="151'
</body>

</html>
```

Z-index

Z-index可被用于将在一个元素放置于另一元素之后。

```
<html>
<head>
<style type="text/css">
img.x
{
position:absolute;
left:0px;
top:0px;
z-index:-1
}
</style>
</head>

<body>
<h1>这是一个标题</h1>
<img class="x" src="/i/eg_mouse.jpg" />
<p>默认的 z-index 是 0。Z-index -1 拥有更低的优先级。</p>
</body>

</html>
```

Z-index

上面的例子中的元素已经更改了Z-index。

```
<html>
<head>
<style type="text/css">
img.x
{
position:absolute;
left:0px;
top:0px;
z-index:1
}
</style>
</head>

<body>
<h1>这是一个标题</h1>
<img class="x" src="/i/eg_mouse.jpg" />
<p>默认的 z-index 是 0。Z-index 1 拥有更高的优先级。</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：position 属性

CSS right 属性

实例

把图像的右边缘设置在其包含元素右边缘向左 5 像素的位置：

```
img
  {
  position:absolute;
  right:5px;
  }
```

浏览器支持

所有主流浏览器都支持 right 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

right 属性规定元素的右边缘。该属性定义了定位元素右外边距边界与其包含块右边界之间的偏移。

注释：如果 "position" 属性的值为 "static"，那么设置 "right" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，left 的计算值始终等于 right。

默认值：	auto
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.right="50px"

可能的值

值	描述
auto	默认值。通过浏览器计算右边缘的位置。
%	设置以包含元素的百分比计的右边位置。可使用负值。
length	使用 px、cm 等单位设置元素的右边位置。可使用负值。
inherit	规定应该从父元素继承 right 属性的值。

TIY 实例

使用固定值设置图像的右边缘

本例演示如何使用固定值设置图像的右边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
right:0px
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

使用百分比设置图像的右边缘

本例演示如何使用百分比值设置图像的右边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
right:5%
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：right 属性

CSS top 属性

实例

把图像的上边缘设置在其包含元素上边缘之下 5 像素高的位置：

```
img
  {
  position:absolute;
  top:5px;
  }
```

浏览器支持

所有主流浏览器都支持 top 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

top 属性规定元素的顶部边缘。该属性定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移。

注释：如果 "position" 属性的值为 "static"，那么设置 "top" 属性不会产生任何效果。

说明

对于 static 元素，为 auto；对于长度值，则为相应的绝对长度；对于百分比数值，为指定值；否则为 auto。

对于相对定义元素，如果 top 和 bottom 都是 auto，其计算值则都是 0；如果其中之一为 auto，则取另一个值的相反数；如果二者都不是 auto，bottom 将取 top 值的相反数。

默认值：	auto
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.top="50px"

可能的值

值	描述
auto	默认值。通过浏览器计算上边缘的位置。
%	设置以包含元素的百分比计的上边位置。可使用负值。
length	使用 px、cm 等单位设置元素的上边位置。可使用负值。
inherit	规定应该从父元素继承 top 属性的值。

TIY 实例

使用固定值设置图像的上边缘

本例演示如何使用固定值设置图像的上边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
top:0px
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

使用百分比设置图像的上边缘

本例演示如何使用百分比值设置图像的上边缘。

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
top:5%
}
</style>
</head>
<body>

<h1>这是标题</h1>
<img class="normal" src="/i/eg_smile.gif" />
<p>一些文本。一些文本。一些文本。一些文本。一些文本。一些文本。</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：top 属性

CSS vertical-align 属性

实例

垂直对齐一幅图像：

```
img
  {
  vertical-align:text-top;
  }
```

浏览器支持

所有浏览器都支持 vertical-align 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

vertical-align 属性设置元素的垂直对齐方式。

说明

该属性定义行内元素的基线相对于该元素所在行的基线的垂直对齐。允许指定负长度值和百分比值。这会使元素降低而不是升高。在表单元格中，这个属性会设置单元格框中的单元格内容的对齐方式。

默认值：	baseline
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.verticalAlign="bottom"

可能的值

值	描述
baseline	默认。元素放置在父元素的基线上。
sub	垂直对齐文本的下标。
super	垂直对齐文本的上标
top	把元素的顶端与行中最高元素的顶端对齐
text-top	把元素的顶端与父元素字体的顶端对齐
middle	把此元素放置在父元素的中部。
bottom	把元素的顶端与行中最低的元素的顶端对齐。
text-bottom	把元素的底端与父元素字体的底端对齐。
length	
%	使用 "line-height" 属性的百分比值来排列此元素。允许使用负值。
inherit	规定应该从父元素继承 vertical-align 属性的值。

TIY 实例

垂直对齐图象

本例演示如何在文本中垂直排列图象。

```
<html>

<head>
<style type="text/css">
img.top {vertical-align:text-top}
img.bottom {vertical-align:text-bottom}
</style>
</head>

<body>

<p>
这是一幅<img class="top" border="0" src="/i/eg_cute.gif" />位于段落中
</p>

<p>
这是一幅<img class="bottom" border="0" src="/i/eg_cute.gif" />位于段
</p>

</body>

</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：verticalAlign 属性

CSS visibility 属性

实例

使 h2 元素不可见：

```
h2
  {
  visibility:hidden;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 visibility 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持 "inherit" 和 "collapse" 属性值。

定义和用法

visibility 属性规定元素是否可见。

提示：即使不可见的元素也会占据页面上的空间。请使用 "display" 属性来创建不占据页面空间的不可见元素。

说明

这个属性指定是否显示一个元素生成的元素框。这意味着元素仍占据其本来的空间，不过可以完全不可见。值 collapse 在表中用于从表布局中删除列或行。

默认值：	visible
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.visibility="hidden"

可能的值

值	描述
visible	默认值。元素是可见的。
hidden	元素是不可见的。
collapse	当在表格元素中使用时，此值可删除一行或一列，但是它不会影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他的元素上，会呈现为 "hidden"。
inherit	规定应该从父元素继承 visibility 属性的值。

TIY 实例

如何使元素不可见

本例演示如何使元素不可见。你希望元素被显示出来，还是不呢？

```
<html>
<head>
<style type="text/css">
h1.visible {visibility:visible}
h1.invisible {visibility:hidden}
</style>
</head>

<body>
<h1 class="visible">这是可见的标题</h1>
<h1 class="invisible">这是不可见的标题</h1>
</body>

</html>
```

把表格元素设置为 collapse

本例演示如何使表格元素叠加？

```
<html>
<head>
<style type="text/css">
tr.coll
 {
 visibility:collapse
 }
</style>
</head>
<body>

<table border="1">
<tr>
<td>Adams</td>
<td>John</td>
</tr>
<tr class="coll">
<td>Bush</td>
<td>George</td>
</tr>
</table>

</body>
</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：visibility 属性

CSS z-index 属性

实例

设置图像的 z-index：

```
img
  {
  position:absolute;
  left:0px;
  top:0px;
  z-index:-1;
  }
```

浏览器支持

所有主流浏览器都支持 z-index 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

z-index 属性设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。

注释：元素可拥有负的 z-index 属性值。

注释：Z-index 仅能在定位元素上奏效（例如 position:absolute;）！

说明

该属性设置一个定位元素沿 z 轴的位置，z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远。

默认值：	auto
继承性：	no
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.zIndex="1"
---	---

可能的值

值	描述
auto	默认。堆叠顺序与父元素相等。
number	设置元素的堆叠顺序。
inherit	规定应该从父元素继承 z-index 属性的值。

TIY 实例

Z-index

Z-index 可用于将在一个元素放置于另一元素之后。

```
<html>
<head>
<style type="text/css">
img.x
{
position:absolute;
left:0px;
top:0px;
z-index:-1
}
</style>
</head>

<body>
<h1>这是一个标题</h1>
<img class="x" src="/i/eg_mouse.jpg" />
<p>默认的 z-index 是 0。Z-index -1 拥有更低的优先级。</p>
</body>

</html>
```

Z-index

上例中的元素已经更改了 Z-index。

```
<html>
<head>
<style type="text/css">
img.x
{
position:absolute;
left:0px;
top:0px;
z-index:1
}
</style>
</head>

<body>
<h1>这是一个标题</h1>
<img class="x" src="/i/eg_mouse.jpg" />
<p>默认的 z-index 是 0。Z-index 1 拥有更高的优先级。</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 定位

HTML DOM 参考手册：zIndex 属性

CSS 打印属性（Print）

属性	描述	CSS
orphans	设置当元素内部发生分页时必须在页面底部保留的最少行数。	2
page-break-after	设置元素后的分页行为。	2
page-break-before	设置元素前的分页行为。	2
page-break-inside	设置元素内部的分页行为。	2
widows	设置当元素内部发生分页时必须在页面顶部保留的最少行数。	2

CSS page-break-after 属性

实例

设置在表格元素之后始终进行分页的分页行为：

```
<html>
<head>
<style>
@media print
{
table {page-break-after:always;}
}
</style>
</head>

<body>
....
</body>
</html>
```

浏览器支持

所有浏览器都支持 page-break-after 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "left"、"right" 以及 "inherit"。

注释：Firefox、Chrome 以及 Safari 不支持属性值 "avoid"、"left" 以及 "right"。

定义和用法

page-break-after 属性设置元素后的 page-breaking 行为。

尽管可以用 always 强制放上分页符，但是无法保证避免分页符的插入，创作人员最多只能要求用户代理理尽可能避免插入分页。

应用于：position 值为 relative 或 static 的非浮动块级元素。

注释：请尽可能少地使用分页属性，并且避免在表格、浮动元素、带有边框的块元素中使用分页属性。

默认值：	**auto**
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.pageBreakAfter="always"

可能的值

值	描述
auto	默认。如果必要则在元素后插入分页符。
always	在元素后插入分页符。
avoid	避免在元素后插入分页符。
left	在元素之后足够的分页符，一直到一张空白的左页为止。
right	在元素之后足够的分页符，一直到一张空白的右页为止。
inherit	规定应该从父元素继承 page-break-after 属性的设置。

相关页面

HTML DOM 参考手册：pageBreakAfter 属性

CSS page-break-before 属性

实例

设置在表格元素之后始终进行分页的分页行为：

```
<html>
<head>
<style>
@media print
{
table {page-break-before:always;}
}
</style>
</head>

<body>
....
</body>
</html>
```

浏览器支持

所有浏览器都支持 page-break-before 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "left"、"right" 以及 "inherit"。

注释：Firefox、Chrome 以及 Safari 不支持属性值 "avoid"、"left" 以及 "right"。

定义和用法

page-break-before 属性设置元素前的 page-breaking 行为。

尽管可以用 always 强制放上分页符，但是无法保证避免分页符的插入，创作人员最多只能要求用户代理尽可能避免插入分页。

应用于：position 值为 relative 或 static 的非浮动块级元素。

注释：请尽可能少地使用分页属性，并且避免在表格、浮动元素、带有边框的块元素中使用分页属性。

默认值：	auto
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.pageBreakBefore="always"

可能的值

值	描述
auto	默认值。如果必要则在元素前插入分页符。
always	在元素前插入分页符。
avoid	避免在元素前插入分页符。
left	在元素之前足够的分页符，一直到一张空白的左页为止。
right	在元素之前足够的分页符，一直到一张空白的右页为止。
inherit	规定应该从父元素继承 page-break-before 属性的设置。

相关页面

HTML DOM 参考手册：pageBreakBefore 属性

CSS page-break-inside 属性

实例

设置在表格元素内部避免进行分页的分页行为：

```
<html>
<head>
<style>
@media print
{
table {page-break-inside:avoid;}
}
</style>
</head>

<body>
....
</body>
</html>
```

浏览器支持

只有 Opera 浏览器支持 page-break-inside 属性。

定义和用法

page-break-inside 属性设置元素内部的 page-breaking 行为。

尽管可以用 always 强制放上分页符，但是无法保证避免分页符的插入，创作人员最多只能要求用户代理理尽可能避免插入分页。

应用于：position 值为 relative 或 static 的非浮动块级元素。

注释：请尽可能少地使用分页属性，并且避免在表格、浮动元素、带有边框的块元素中使用分页属性。

默认值：	**auto**
继承性：	no
版本：	CSS2
JavaScript 语法：	*object*.style.pageBreakInside="avoid"

可能的值

值	描述
auto	默认。如果必要则在元素内部插入分页符。
avoid	避免在元素内部插入分页符。
inherit	规定应该从父元素继承 page-break-inside 属性的设置。

相关页面

HTML DOM 参考手册：pageBreakInside 属性

CSS 表格属性（Table）

属性	描述	CSS
border-collapse	规定是否合并表格边框。	2
border-spacing	规定相邻单元格边框之间的距离。	2
caption-side	规定表格标题的位置。	2
empty-cells	规定是否显示表格中的空单元格上的边框和背景。	2
table-layout	设置用于表格的布局算法。	2

CSS border-collapse 属性

实例

为表格设置合并边框模型：

```
table
  {
  border-collapse:collapse;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 border-collapse 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

注释：如果没有规定 !DOCTYPE，则 border-collapse 可能产生意想不到的结果。

定义和用法

border-collapse 属性设置表格的边框是否被合并为一个单一的边框，还是象在标准的 HTML 中那样分开显示。

默认值：	separate
继承性：	yes
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.borderCollapse="collapse"
---	---

可能的值

值	描述
separate	默认值。边框会被分开。不会忽略 border-spacing 和 empty-cells 属性。
collapse	如果可能，边框会合并为一个单一的边框。会忽略 border-spacing 和 empty-cells 属性。
inherit	规定应该从父元素继承 border-collapse 属性的值。

TIY 实例

合并表格边框

本例演示是否把表格边框显示为一条单独的边框，还是像标准的 HTML 中那样分开显示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt
<html>
<head>
<style type="text/css">
table
  {
  border-collapse:collapse;
  }

table, td, th
  {
  border:1px solid black;
  }
</style>
</head>

<body>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>Bill</td>
<td>Gates</td>
</tr>
<tr>
<td>Steven</td>
<td>Jobs</td>
</tr>
</table>
<p><b>注释：</b>如果没有规定 !DOCTYPE，border-collapse 属性可能会引起意想
</body>
</html>
```

相关页面

CSS 教程：CSS 表格

HTML DOM 参考手册：borderCollapse 属性

CSS border-spacing 属性

实例

为表格设置 border-spacing：

```
table
  {
  border-collapse:separate;
  border-spacing:10px 50px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 border-spacing 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 border-spacing 属性。

定义和用法

border-spacing 属性设置相邻单元格的边框间的距离（仅用于“边框分离”模式）。

注释：某些版本的IE浏览器不支持此属性。

说明

该属性指定分隔边框模型中单元格边界之间的距离。在指定的两个长度值中，第一个是水平间隔，第二个是垂直间隔。除非 border-collapse 被设置为 separate，否则将忽略这个属性。尽管这个属性只应用于表，不过它可以由表中的所有元素继承。

默认值：	***not specified***
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.borderSpacing="15px"

可能的值

值	描述
length length	

规定相邻单元的边框之间的距离。使用 px、cm 等单位。不允许使用负值。

如果定义一个 *length* 参数，那么定义的是水平和垂直间距。

如果定义两个 *length* 参数，那么第一个设置水平间距，而第二个设置垂直间距。

| | inherit | 规定应该从父元素继承 border-spacing 属性的值。 |

TIY 实例

设置表格边框之间的空白

本例演示如何设置单元格边框之间的距离。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt

<html>
<head>
<style type="text/css">
table.one
{
border-collapse: separate;
border-spacing: 10px
}
table.two
{
border-collapse: separate;
border-spacing: 10px 50px
}
</style>
</head>
<body>

<table class="one" border="1">
<tr>
<td>Adams</td>
<td>John</td>
</tr>
<tr>
<td>Bush</td>
<td>George</td>
</tr>
</table>

<br />

<table class="two" border="1">
<tr>
<td>Carter</td>
<td>Thomas</td>
</tr>
<tr>
<td>Gates</td>
<td>Bill</td>
</tr>
</table>

<p><b>注释：</b>如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高

</body>
</html>
```

相关页面

CSS 教程：CSS 表格

HTML DOM 参考手册：borderSpacing 属性

CSS caption-side 属性

实例

规定表格标题的放置方式：

```
caption
  {
  caption-side:bottom;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 caption-side 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 caption-side 属性。

定义和用法

caption-side 属性设置表格标题的位置。

说明

该属性指定了表标题相对于表框的放置位置。表标题显示为好像它是表之前（或之后）的一个块级元素。

默认值：	top
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.captionSide="bottom"

可能的值

值	描述
top	默认值。把表格标题定位在表格之上。
bottom	把表格标题定位在表格之下。
inherit	规定应该从父元素继承 caption-side 属性的值。

TIY 实例

设置表格标题的位置

本例演示如何定位表格的标题。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "htt

<html>
<head>
<style type="text/css">
caption
{
caption-side:bottom
}
</style>
</head>
<body>

<table border="1">
<caption>This is a caption</caption>
<tr>
<td>Adams</td>
<td>John</td>
</tr>
<tr>
<td>Bush</td>
<td>George</td>
</tr>
</table>

<p><b>注释：</b>如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高

</body>
</html>
```

相关页面

CSS 教程：CSS 表格

HTML DOM 参考手册：captionSide 属性

CSS empty-cells 属性

实例

隐藏表格中空单元格上的边框和背景：

```
table
  {
  border-collapse:separate;
  empty-cells:hide;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 empty-cells 属性。

注释：如果已规定 !DOCTYPE，那么 Internet Explorer 8 （以及更高版本）支持 empty-cells 属性。

定义和用法

empty-cells 属性设置是否显示表格中的空单元格（仅用于“分离边框”模式）。

注释：某些版本的 IE 浏览器不支持此属性。

说明

该属性定义了不包含任何内容的表单元格如何表示。如果显示，就会绘制出单元格的边框和背景。除非 border-collapse 设置为 separate，否则将忽略这个属性。

默认值：	show
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.emptyCells="hide"

可能的值

值	描述
hide	不在空单元格周围绘制边框。
show	在空单元格周围绘制边框。默认。
inherit	规定应该从父元素继承 empty-cells 属性的值。

相关页面

CSS 教程：CSS 表格

HTML DOM 参考手册：emptyCells 属性

CSS table-layout 属性

实例

设置表格布局算法：

```
table
  {
  table-layout:fixed;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 table-layout 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

tableLayout 属性用来显示表格单元格、行、列的算法规则。

固定表格布局：

固定表格布局与自动表格布局相比，允许浏览器更快地对表格进行布局。

在固定表格布局中，水平布局仅取决于表格宽度、列宽度、表格边框宽度、单元格间距，而与单元格的内容无关。

通过使用固定表格布局，用户代理在接收到第一行后就可以显示表格。

自动表格布局：

在自动表格布局中，列的宽度是由列单元格中没有折行的最宽的内容设定的。

此算法有时会较慢，这是由于它需要在确定最终的布局之前访问表格中所有的内容。

说明

该属性指定了完成表布局时所用的布局算法。固定布局算法比较快，但是不太灵活，而自动算法比较慢，不过更能反映传统的 HTML 表。

默认值：	auto
继承性：	yes
---	---
版本：	CSS2
---	---
JavaScript 语法：	*object*.style.tableLayout="fixed"
---	---

可能的值

值	描述
automatic	默认。列宽度由单元格内容设定。
fixed	列宽由表格宽度和列宽度设定。
inherit	规定应该从父元素继承 table-layout 属性的值。

TIY 实例

设置表格的布局

本例演示如何设置表格的布局。

```
<html>
<head>
<style type="text/css">
table.one
{
table-layout: automatic
}
table.two
{
table-layout: fixed
}
</style>
</head>
<body>

<table class="one" border="1" width="100%">
<tr>
<td width="20%">1000000000000000000000000000</td>
<td width="40%">10000000</td>
<td width="40%">100</td>
</tr>
</table>

<br />

<table class="two" border="1" width="100%">
<tr>
<td width="20%">1000000000000000000000000000</td>
<td width="40%">10000000</td>
<td width="40%">100</td>
</tr>
</table>

</body>
</html>
```

相关页面

CSS 教程：CSS 表格

HTML DOM 参考手册：tableLayout 属性

CSS 文本属性（Text）

属性	描述	CSS
color	设置文本的颜色。	1
direction	规定文本的方向 / 书写方向。	2
letter-spacing	设置字符间距。	1
line-height	设置行高。	1
text-align	规定文本的水平对齐方式。	1
text-decoration	规定添加到文本的装饰效果。	1
text-indent	规定文本块首行的缩进。	1
text-shadow	规定添加到文本的阴影效果。	2
text-transform	控制文本的大小写。	1
unicode-bidi	设置文本方向。	2
white-space	规定如何处理元素中的空白。	1
word-spacing	设置单词间距。	1
hanging-punctuation	规定标点字符是否位于线框之外。	3
punctuation-trim	规定是否对标点字符进行修剪。	3
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行。	3
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色。	3
text-justify	规定当 text-align 设置为 "justify" 时所使用的对齐方法。	3
text-outline	规定文本的轮廓。	3
text-overflow	规定当文本溢出包含元素时发生的事情。	3
text-shadow	向文本添加阴影。	3
text-wrap	规定文本的换行规则。	3
word-break	规定非中日韩文本的换行规则。	3
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行。	3

CSS color 属性

实例

为不同元素设置文本颜色：

```
body
  {
  color:red;
  }
h1
  {
  color:#00ff00;
  }
p
  {
  color:rgb(0,0,255);
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 color 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

color 属性规定文本的颜色。

这个属性设置了一个元素的前景色（在 HTML 表现中，就是元素文本的颜色）；光栅图像不受 color 影响。这个颜色还会应用到元素的所有边框，除非被 border-color 或另外某个边框颜色属性覆盖。

要设置一个元素的前景色，最容易的方法是使用 color 属性。

默认值：	***not specified***
继承性：	yes
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.color="#FF0000"
---	---

提示和注释

提示：请使用合理的背景颜色和文本颜色搭配，这样可以提高文本的可读性。

可能的值

值	描述
color_name	规定颜色值为颜色名称的颜色（比如 red）。
hex_number	规定颜色值为十六进制值的颜色（比如 #ff0000）。
rgb_number	规定颜色值为 rgb 代码的颜色（比如 rgb(255,0,0)）。
inherit	规定应该从父元素继承颜色。

TIY 实例

设置文本颜色

本例演示如何设置文本的颜色。

```
<html>
<head>
<style type="text/css">
body {color:red}
h1 {color:#00ff00}
p.ex {color:rgb(0,0,255)}
</style>
</head>

<body>
<h1>这是 heading 1</h1>
<p>这是一段普通的段落。请注意，该段落的文本是红色的。在 body 选择器中定义了本页
<p class="ex">该段落定义了 class="ex"。该段落中的文本是蓝色的。</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：color 属性

CSS direction 属性

实例

把文本方向设置为“从右向左”：

```
div
  {
  direction: rtl
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 direction 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

direction 属性规定文本的方向 / 书写方向。

该属性指定了块的基本书写方向，以及针对 Unicode 双向算法的嵌入和覆盖方向。不支持双向文本的用户代理可以忽略这个属性。

默认值：	ltr
继承性：	yes
版本：	CSS2
JavaScript 语法：	*object*.style.direction="rtl"

可能的值

值	描述
ltr	默认。文本方向从左到右。
rtl	文本方向从右到左。
inherit	规定应该从父元素继承 direction 属性的值。

TIY 实例

设置文本的方向

本例演示如何设置文本的方向。

```
<html>
<head>
<style type="text/css">
div.one
{
direction: rtl
}
div.two
{
direction: ltr
}
</style>
</head>
<body>

<div class="one">Some text. Right-to-left direction.</div>
<div class="two">Some text. Left-to-right direction.</div>

</body>
</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：direction 属性

CSS letter-spacing 属性

实例

设置 h1 和 h2 元素的字母间距：

```
h1 {letter-spacing:2px}
h2 {letter-spacing:-3px}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 letter-spacing 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

letter-spacing 属性增加或减少字符间的空白（字符间距）。

该属性定义了在文本字符框之间插入多少空间。由于字符字形通常比其字符框要窄，指定长度值时，会调整字母之间通常的间隔。因此，normal 就相当于值为 0。

注释：允许使用负值，这会让字母之间挤得更紧。

默认值：	normal
继承性：	yes
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.letterSpacing="3px"
---	---

可能的值

值	描述
normal	默认。规定字符间没有额外的空间。
length	定义字符间的固定空间（允许使用负值）。
inherit	规定应该从父元素继承 letter-spacing 属性的值。

TIY 实例

规定字符间距（字母间隔）

本例演示如何增加或减少字符间距。

```
<html>

<head>
<style type="text/css">
h1 {letter-spacing: -0.5em}
h4 {letter-spacing: 20px}
</style>
</head>

<body>
<h1>This is header 1</h1>
<h4>This is header 4</h4>
</body>

</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：letterSpacing 属性

CSS line-height 属性

实例

设置以百分比计的行高：

```
p.small {line-height:90%}
p.big {line-height:200%}
```

（可以在页面底部查看更多实例）

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 line-height 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

line-height 属性设置行间的距离（行高）。

注释：不允许使用负值。

说明

该属性会影响行框的布局。在应用到一个块级元素时，它定义了该元素中基线之间的最小距离而不是最大距离。

line-height 与 font-size 的计算值之差（在 CSS 中成为“行间距”）分为两半，分别加到一个文本行内容的顶部和底部。可以包含这些内容的最小框就是行框。

原始数字值指定了一个缩放因子，后代元素会继承这个缩放因子而不是计算值。

默认值：	**normal**
继承性：	yes
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.lineHeight="2"
---	---

可能的值

值	描述
normal	默认。设置合理的行间距。
number	设置数字，此数字会与当前的字体尺寸相乘来设置行间距。
length	设置固定的行间距。
%	基于当前字体尺寸的百分比行间距。
inherit	规定应该从父元素继承 line-height 属性的值。

TIY 实例

使用百分比设置行间距

本例演示如何使用百分比值来设置段落中的行间距。

```
<html>

<head>
<style type="text/css">
p.small {line-height: 90%}
p.big {line-height: 200%}
</style>
</head>

<body>

<p>
这是拥有标准行高的段落。
在大多数浏览器中默认行高大约是 110% 到 120%。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
</p>

<p class="small">
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
</p>

<p class="big">
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
</p>

</body>
</html>
```

使用像素值设置行间距

本例演示如何使用像素值来设置段落中的行间距。

```
<html>

<head>
<style type="text/css">
p.small
  {
  line-height: 10px
  }
p.big
  {
  line-height: 30px
  }
</style>
</head>

<body>

<p>
这是拥有标准行高的段落。
在大多数浏览器中默认行高大约是 20px。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
</p>

<p class="small">
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
</p>

<p class="big">
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
</p>

</body>
</html>
```

使用数值来设置行间距

本例演示如何使用一个数值来设置段落中的行间距。

```
<html>

<head>
<style type="text/css">
p.small
{
line-height: 0.5
}
p.big
{
line-height: 2
}
</style>
</head>

<body>

<p>
这是拥有标准行高的段落。
默认行高大约是 1。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
</p>

<p class="small">
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
</p>

<p class="big">
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：lineHeight 属性

CSS text-align 属性

实例

设置 h1、h2、h3 元素的文本对齐方式：

```
h1 {text-align:center}
h2 {text-align:left}
h3 {text-align:right}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 text-align 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

text-align 属性规定元素中的文本的水平对齐方式。

该属性通过指定行框与哪个点对齐，从而设置块级元素内文本的水平对齐方式。通过允许用户代理调整行内容中字母和字之间的间隔，可以支持值 justify；不同用户代理可能会得到不同的结果。

默认值：	如果 **direction** 属性是 **ltr**，则默认值是 **left**；如果 **direction** 是 **rtl**，则为 **right**。
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.textAlign="right"

可能的值

值	描述
left	把文本排列到左边。默认值：由浏览器决定。
right	把文本排列到右边。
center	把文本排列到中间。
justify	实现两端对齐文本效果。
inherit	规定应该从父元素继承 text-align 属性的值。

值 justify

最后一个水平对齐属性是 justify，它会带来自己的一些问题。

值 justify 可以使文本的两端都对齐。在两端对齐文本中，文本行的左右两端都放在父元素的内边界上。然后，调整单词和字母间的间隔，使各行的长度恰好相等。您也许已经注意到了，两端对齐文本在打印领域很常见。不过在 CSS 中，还需要多做些考虑。

要由用户代理（而不是 CSS）来确定两端对齐文本如何拉伸，以填满父元素左右边界之间的空间。例如，有些浏览器可能只在单词之间增加额外的空间，而另外一些浏览器可能会平均分布字母间的额外空间（不过 CSS 规范特别指出，如果 letter-spacing 属性指定为一个长度值，“用户代理不能进一步增加或减少字符间的空间”）。还有一些用户代理可能会减少某些行的空间，使文本挤得更紧密。所有这些做法都会影响元素的外观，甚至改变其高度，这取决于用户代理的对齐选择影响了多少文本行。

CSS 也没有指定应当如何处理连字符（注1）。大多数两端对齐文本都使用连字符将长单词分开放在两行上，从而缩小单词之间的间隔，改善文本行的外观。不过，由于 CSS 没有定义连字符行为，用户代理不太可能自动加连字符。因此，在 CSS 中，两端对齐文本看上去没有打印出来好看，特别是元素可能太窄，以至于每行只能放下几个单词。当然，使用窄设计元素是可以的，不过要当心相应的缺点。

注1：CSS 中没有说明如何处理连字符，因为不同的语言有不同的连字符规则。规范没有尝试去调和这样一些很可能不完备的规则，而是干脆不提这个问题。

TIY 实例

对齐文本

本例演示如何对齐文本。

```
<html>
<head>
<style type="text/css">
h1 {text-align: center}
h2 {text-align: left}
h3 {text-align: right}
</style>
</head>

<body>
<h1>这是标题 1</h1>
<h2>这是标题 2</h2>
<h3>这是标题 3</h3>
</body>

</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：textAlign 属性

CSS text-decoration 属性

实例

设置 h1、h2、h3、h4 元素的文本修饰：

```
h1 {text-decoration:overline}
h2 {text-decoration:line-through}
h3 {text-decoration:underline}
h4 {text-decoration:blink}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 text-decoration 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

注释：IE、Chrome 或 Safari 不支持 "blink" 属性值。

定义和用法

text-decoration 属性规定添加到文本的修饰。

注释：修饰的颜色由 "color" 属性设置。

说明

这个属性允许对文本设置某种效果，如加下划线。如果后代元素没有自己的装饰，祖先元素上设置的装饰会“延伸”到后代元素中。不要求用户代理支持 blink。

默认值：	**none**
继承性：	no
版本：	CSS1
JavaScript 语法：	*object*.style.textDecoration="overline"

可能的值

值	描述
none	默认。定义标准的文本。
underline	定义文本下的一条线。
overline	定义文本上的一条线。
line-through	定义穿过文本下的一条线。
blink	定义闪烁的文本。
inherit	规定应该从父元素继承 text-decoration 属性的值。

TIY 实例

修饰文本

本例演示如何向文本添加修饰。

```
<html>
<head>
<style type="text/css">
h1 {text-decoration: overline}
h2 {text-decoration: line-through}
h3 {text-decoration: underline}
h4 {text-decoration:blink}
a {text-decoration: none}
</style>
</head>

<body>
<h1>这是标题 1</h1>
<h2>这是标题 2</h2>
<h3>这是标题 3</h3>
<h4>这是标题 4</h4>
<p><a href="http://www.w3school.com.cn/index.html">这是一个链接</a></
</body>

</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：textDecoration 属性

CSS text-indent 属性

实例

将段落的第一行缩进 50 像素：

```
p
  {
  text-indent:50px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 text-indent 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义

text-indent 属性规定文本块中首行文本的缩进。

注释：允许使用负值。如果使用负值，那么首行会被缩进到左边。

注意：在 CSS 2.1 之前，text-indent 总是继承计算值，而不是声明值。

说明

用于定义块级元素中第一个内容行的缩进。这最常用于建立一个“标签页”效果。允许指定负值，这会产生一种“悬挂缩进”的效果。

默认值：	***not specified***
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.textIndent="50px"

可能的值

值	描述
length	定义固定的缩进。默认值：0。
%	定义基于父元素宽度的百分比的缩进。
inherit	规定应该从父元素继承 text-indent 属性的值。

TIY 实例

缩进文本

本例演示如何缩进文本首行。

```
<html>
<head>
<style type="text/css">
p {text-indent: 1cm}
</style>
</head>

<body>
<p>
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
这是段落中的一些文本。
</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：textIndent 属性

CSS text-transform 属性

实例

转换不同元素中的文本：

```
h1 {text-transform:uppercase}
h2 {text-transform:capitalize}
p {text-transform:lowercase}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 text-transform 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

text-transform 属性控制文本的大小写。

说明

这个属性会改变元素中的字母大小写，而不论源文档中文本的大小写。如果值为 capitalize，则要对某些字母大写，但是并没有明确定义如何确定哪些字母要大写，这取决于用户代理如何识别出各个“词”。

默认值：	none
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.textTransform="uppercase"

提示和注释

注释：不同的用户代理可能会用不同的方法来确定单词从哪里开始，相应地确定哪些字母要大写。例如，文本 "w3-school" 可以用两种方式显示："W3-school" 和 "W3-School"。CSS 并没有规定哪一种是正确的，所以这两种都是可以的。

可能的值

值	描述
none	默认。定义带有小写字母和大写字母的标准的文本。
capitalize	文本中的每个单词以大写字母开头。
uppercase	定义仅有大写字母。
lowercase	定义无大写字母，仅有小写字母。
inherit	规定应该从父元素继承 text-transform 属性的值。

TIY 实例

控制文本中的字母

本例演示如何控制文本中的字母的大小写。

```
<html>

<head>
<style type="text/css">
  h1 {text-transform: uppercase}
  p.uppercase {text-transform: uppercase}
  p.lowercase {text-transform: lowercase}
  p.capitalize {text-transform: capitalize}
</style>
</head>

<body>
<h1>This Is An H1 Element</h1>
<p class="uppercase">This is some text in a paragraph.</p>
<p class="lowercase">This is some text in a paragraph.</p>
<p class="capitalize">This is some text in a paragraph.</p>
</body>

</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：textTransform 属性

CSS unicode-bidi 属性

定义

unicode-bidi 属性设置文本的方向。

继承性：Yes

说明

尽管 CSS 试图处理书写方向，但 Unicode 有一种更健壮的方式来处理方向性。利用属性 unicode-bidi，CSS 创作人员可以充分利用 Unicode 的某些功能。

可能的值

在这里，我们将简要引用 CSS2.1 规范中关于这些值的描述，这些描述很好地说明了各个值的实现。

normal

元素不会对双向算法打开附加的一层嵌套。对于行内元素，顺序的隐式重排会跨元素边界进行。

embed

如果是一个行内元素，这个值对于双向算法会打开附件的一层嵌套。这个嵌套层的方向由 direction 属性指定。会在元素内部隐式地完成顺序重排。这对应于在元素开始处增加一个 LRE（对于 direction:ltr ：U+202A）或 RLE（对于 direction:rtl ：U+202B），并在元素的最后增加一个 PDF（U+202C）。

bidi-override

这会为行内元素创建一个覆盖。对于块级元素，将为不在另一块中的行内后代创建一个覆盖。这说明，顺序重排在元素内部严格按照 direction 属性进行；忽略了双向算法的隐式部分。这对应于在元素开始处增加一个 LRO（对于 direction:ltr ：U+202D）或 RLO（对于 direction:rtl ：U+202E），并在元素最后增加一个 PDF（U+202C）。

CSS white-space 属性

实例

规定段落中的文本不进行换行：

```
p
  {
  white-space: nowrap
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 white-space 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义和用法

white-space 属性设置如何处理元素内的空白。

这个属性声明建立布局过程中如何处理元素中的空白符。值 pre-wrap 和 pre-line 是 CSS 2.1 中新增的。

默认值：	normal
继承性：	yes
版本：	CSS1
JavaScript 语法：	*object*.style.whiteSpace="pre"

可能的值

值	描述
normal	默认。空白会被浏览器忽略。
pre	空白会被浏览器保留。其行为方式类似 HTML 中的 <pre> 标签。
nowrap	文本不会换行，文本会在在同一行上继续，直到遇到 标签为止。
pre-wrap	保留空白符序列，但是正常地进行换行。
pre-line	合并空白符序列，但是保留换行符。
inherit	规定应该从父元素继承 white-space 属性的值。

TIY 实例

在元素中禁止文本折行

本例演示如何禁止在元素中的文本折行。

```
<html>
<head>
<style type="text/css">
p
{
white-space: nowrap
}
</style>
</head>
<body>

<p>
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
这是一些文本。
</p>

</body>
</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：whiteSpace 属性

CSS word-spacing 属性

实例

规定段落中的字间距是 25 像素：

```
p
  {
  word-spacing:25px;
  }
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有浏览器都支持 word-spacing 属性。

注释：任何的版本的 Internet Explorer （包括 IE8）都不支持属性值 "inherit"。

定义

word-spacing 属性增加或减少单词间的空白（即字间隔）。

该属性定义元素中字之间插入多少空白符。针对这个属性，“字” 定义为由空白符包围的一个字符串。如果指定为长度值，会调整字之间的通常间隔；所以，normal 就等同于设置为 0。允许指定负长度值，这会让字之间挤得更紧。

注释：允许使用负值。

默认值：	normal
继承性：	yes
---	---
版本：	CSS1
---	---
JavaScript 语法：	*object*.style.wordSpacing="10px"
---	---

提示和注释

注释：CSS 把"字（word）"定义为任何非空白符字符组成的串，并由某种空白字符包围。这个定义没有实际的语义，它只是假设一个文档包含由一个或多个空白字符包围的字。支持 CSS 的用户代理不一定能确定一个给定语言中哪些是合法的字，而哪些不是。尽管这个定义没有多大价值，不过它意味着采用象形文字的语言或非罗马书写体往往无法指定字间隔。

提示：利用这个属性，可能会创建字间隔太宽的文档，所以，使用 word-spacing 时要小心。

可能的值

值	描述
normal	默认。定义单词间的标准空间。
length	定义单词间的固定空间。
inherit	规定应该从父元素继承 word-spacing 属性的值。

TIY 实例

增加或减少单词间距（字间隔）

本例演示如何增加段落中单词间的距离。

```
<html>
<head>
<style type="text/css">
p.spread {word-spacing: 30px;}
p.tight {word-spacing: -0.5em;}
</style>
</head>

<body>
<p class="spread">This is some text. This is some text.</p>
<p class="tight">This is some text. This is some text.</p>
</body>
</html>
```

相关页面

CSS 教程：CSS 文本

HTML DOM 参考手册：wordSpacing 属性

CSS3 hanging-punctuation 属性

实例

在 p 元素首行的开始边缘之外放置一个标点符号：

```
p
{
hanging-punctuation:first;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前主流浏览器都不支持 hanging-punctuation 属性。

定义和用法

hanging-punctuation 属性规定把标点符号放在文本整行的开头还是结尾的行框之外。

默认值：	none
继承性：	yes
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.hangingPunctuation="first"
---	---

语法

```
hanging-punctuation: none|first|last|allow-end|force-end;
```

值	描述
none	不在文本整行的开头还是结尾的行框之外放置标签符号。
first	标点附着在首行开始边缘之外。
last	标点附着在首行结尾边缘之外。
allow-end	
force-end	

CSS3 punctuation-trim 属性

实例

修剪 p 元素中每行开头的开启标点：

```
p
{
punctuation-trim:start;
}
```

浏览器支持

目前主流浏览器都不支持 punctuation-trim 属性。

定义和用法

punctuation-trim 属性规定如果标点位于行开头或结尾处，或者临近另一个全角标点符号，是否对标点符号进行修剪。

默认值：	none
继承性：	yes
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.punctuationTrim="start"
---	---

语法

```
punctuation-trim: none|start|end|allow-end|adjacent;
```

值	描述
none	不修剪开启或闭合标点符号。
start	修剪每行结尾的开启标点符号。
end	修剪每行结尾的闭合标点符号。
allow-end	
adjacent	

CSS3 text-emphasis 属性

浏览器支持

目前主流浏览器都不支持 text-emphasis 属性。

定义和用法

text-emphasis 属性是简写属性，用于在一个声明中设置 text-emphasis-style 和 text-emphasis-color。

提示：Adobe 的 "Kenten Generic OpenType Font" 是一个适合重点标记的字体，它专门为重点标记设计。

默认值：	none
继承性：	yes
版本：	CSS3
JavaScript 语法：	*object*.style.textEmphasis="filled blue"

语法

```
text-emphasis: _text-emphasis-style_ _text-emphasis-color_;
```

值	描述
text-emphasis-style	向元素的文本应用重点标记。
text-emphasis-color	定义重点标记的前景色。

CSS3 text-justify 属性

实例

齐行改变单词间的间隔：

```
div
{
text-align:justify;
text-justify:inter-word;
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

只有 Internet Explorer 支持 text-justify 属性。

定义和用法

text-justify 属性规定当 text-align 被设置为 text-align 时的齐行方法。

该属性规定如何对齐行文本进行对齐和分隔。

默认值：	**auto**
继承性：	yes
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.textJustify="inter-word"
---	---

语法

```
text-justify: auto|inter-word|inter-ideograph|inter-cluster|distrib
```

值	描述	测试
auto	浏览器决定齐行算法。	测试
none	禁用齐行。	测试
inter-word	增加/减少单词间的间隔。	测试
inter-ideograph	用表意文本来排齐内容。	测试
inter-cluster	只对不包含内部单词间隔的内容（比如亚洲语系）进行排齐。	测试
distribute	类似报纸版面，除了在东亚语系中最后一行是不齐行的。	测试
kashida	通过拉伸字符来排齐内容。	测试

CSS3 text-outline 属性

实例

设置 text-outline：

```
p.test
{
text-outline: 2px 2px #ff0000;
}
```

浏览器支持

所有主流浏览器都不支持 text-outline 属性。

定义和用法

text-outline 属性规定文本轮廓。

默认值：	**none**
继承性：	yes
版本：	CSS3
JavaScript 语法：	*object*.style.textOutline="2px 2px #ff0000"

语法

```
text-outline: _thickness_ _blur_ _color_;
```

值	描述
thickness	必需。轮廓的粗细。
blur	可选。轮廓的模糊半径。
color	必需。轮廓的颜色。参阅 CSS 颜色值。

CSS3 text-overflow 属性

实例

使用 text-overflow 属性：

```
div.test
{
text-overflow:ellipsis;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 text-overflow 属性。

定义和用法

text-overflow 属性规定当文本溢出包含元素时发生的事情。

默认值：	clip
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.textOverflow="ellipsis"

语法

```
text-overflow: clip|ellipsis|_string_;
```

值	描述	测试
clip	修剪文本。	测试
ellipsis	显示省略符号来代表被修剪的文本。	测试
string	使用给定的字符串来代表被修剪的文本。	

亲自试一试 - 实例

带有 hover 效果的 Text-overflow

本例演示当光标浮动到元素上时如何显示全部文本。

```
<!DOCTYPE html>
<html>
<head>
<style>
div.test
{
white-space:nowrap;
width:12em;
overflow:hidden;
border:1px solid #000000;
}

div.test:hover
{
text-overflow:inherit;
overflow:visible;
}
</style>

</head>

<body>

<p>如果您把光标移动到下面两个 div 上，就能够看到全部文本。</p>

<p>这个 div 使用 "text-overflow:ellipsis" ：</p>

<div class="test" style="text-overflow:ellipsis;">This is some long

<p>这个 div 使用 "text-overflow:clip"：</p>

<div class="test" style="text-overflow:clip;">This is some long tex

</body>
</html>
```

相关页面

CSS3 教程：CSS3 文本效果

CSS3 text-shadow 属性

实例

基础的文本阴影效果：

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 text-shadow 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 text-shadow 属性。

定义和用法

text-shadow 属性向文本设置阴影。

默认值：	none
继承性：	yes
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.textShadow="2px 2px #ff0000"
---	---

语法

```
text-shadow: _h-shadow_ _v-shadow_ _blur_ _color_;
```

注释：text-shadow 属性向文本添加一个或多个阴影。该属性是逗号分隔的阴影列表，每个阴影有两个或三个长度值和一个可选的颜色值进行规定。省略的长度是0。

值	描述	测试
h-shadow	必需。水平阴影的位置。允许负值。	测试
v-shadow	必需。垂直阴影的位置。允许负值。	测试
blur	可选。模糊的距离。	测试
color	可选。阴影的颜色。参阅 CSS 颜色值。	测试

亲自试一试 - 实例

带有模糊效果的文本阴影

该例演示带有模糊效果的文本阴影。

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {text-shadow:2px 2px 8px #FF0000;}
</style>
</head>
<body>

<h1>模糊效果的文本阴影！</h1>

</body>
</html>
```

白色文本上的阴影

本例演示白色文本上的文本阴影。

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
color:white;
text-shadow:2px 2px 4px #000000;
}
</style>
</head>
<body>

<h1>白色文本的阴影效果！</h1>

</body>
</html>
```

霓虹灯效果的文本阴影

本例演示带有霓虹灯效果的文本阴影。

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
text-shadow:0 0 3px #FF0000;
}
</style>
</head>
<body>

<h1>霓虹灯效果的文本阴影！</h1>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 文本效果

CSS3 text-wrap 属性

实例

不允许换行：

```
p.test {text-wrap:none;}
```

浏览器支持

目前主流浏览器都不支持 text-wrap 属性。

定义和用法

text-wrap 属性规定文本的换行（折行）规则。

默认值：	normal
继承性：	yes
版本：	CSS3
JavaScript 语法：	*object*.style.textWrap="none"

语法

```
text-wrap: normal|none|unrestricted|suppress;
```

值	描述
normal	只在允许的换行点进行换行。
none	不换行。元素无法容纳的文本会溢出。
unrestricted	在任意两个字符间换行。
suppress	压缩元素中的换行。浏览器只在行中没有其他有效换行点时进行换行。

CSS3 word-break 属性

实例

在恰当的断字点进行换行：

```
p.test {word-break:hyphenate;}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 word-break 属性。

定义和用法

word-break 属性规定自动换行的处理方法。

提示：通过使用 word-break 属性，可以让浏览器实现在任意位置的换行。

默认值：	normal
继承性：	yes
版本：	CSS3
JavaScript 语法：	*object*.style.wordBreak="keep-all"

语法

```
word-break: normal|break-all|keep-all;
```

值	描述
normal	使用浏览器默认的换行规则。
break-all	允许在单词内换行。
keep-all	只能在半角空格或连字符处换行。

CSS3 word-wrap 属性

实例

允许长单词换行到下一行：

```
p.test {word-wrap:break-word;}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

所有主流浏览器都支持 word-wrap 属性。

定义和用法

word-wrap 属性允许长单词或 URL 地址换行到下一行。

默认值：	normal
继承性：	yes
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.wordWrap="break-word"
---	---

语法

```
word-wrap: normal|break-word;
```

值	描述
normal	只在允许的断字点换行（浏览器保持默认处理）。
break-word	在长单词或 URL 地址内部进行换行。

2D/3D 转换属性（Transform）

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3
transform-style	规定被嵌套元素如何在 3D 空间中显示。	3
perspective	规定 3D 元素的透视效果。	3
perspective-origin	规定 3D 元素的底部位置。	3
backface-visibility	定义元素在不面对屏幕时是否可见。	3

CSS3 transform 属性

实例

旋转 div 元素：

```
div
{
transform:rotate(7deg);
-ms-transform:rotate(7deg); 	/* IE 9 */
-moz-transform:rotate(7deg); 	/* Firefox */
-webkit-transform:rotate(7deg); /* Safari 和 Chrome */
-o-transform:rotate(7deg); 	/* Opera */
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 支持 transform 属性。

Internet Explorer 9 支持替代的 -ms-transform 属性（仅适用于 2D 转换）。

Safari 和 Chrome 支持替代的 -webkit-transform 属性（3D 和 2D 转换）。

Opera 只支持 2D 转换。

定义和用法

transform 属性向元素应用 2D 或 3D 转换。该属性允许我们对元素进行旋转、缩放、移动或倾斜。

为了更好地理解 transform 属性，请查看这个演示。

默认值：	none
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.transform="rotate(7deg)"

语法

```
transform: none|_transform-functions_;
```

值	描述	测试
none	定义不进行转换。	测试
matrix(*n*,*n*,*n*,*n*,*n*,*n*)	定义 2D 转换，使用六个值的矩阵。	测试
matrix3d(*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*)	定义 3D 转换，使用 16 个值的 4x4 矩阵。	
translate(*x*,*y*)	定义 2D 转换。	测试
translate3d(*x*,*y*,*z*)	定义 3D 转换。	
translateX(*x*)	定义转换，只是用 X 轴的值。	测试
translateY(*y*)	定义转换，只是用 Y 轴的值。	测试
translateZ(*z*)	定义 3D 转换，只是用 Z 轴的值。	
scale(*x*,*y*)	定义 2D 缩放转换。	测试
scale3d(*x*,*y*,*z*)	定义 3D 缩放转换。	
scaleX(*x*)	通过设置 X 轴的值来定义缩放转换。	测试

scaleY(*y*)	通过设置 Y 轴的值来定义缩放转换。	测试
scaleZ(*z*)	通过设置 Z 轴的值来定义 3D 缩放转换。	
rotate(*angle*)	定义 2D 旋转，在参数中规定角度。	测试
rotate3d(*x*,*y*,*z*,*angle*)	定义 3D 旋转。	
rotateX(*angle*)	定义沿着 X 轴的 3D 旋转。	测试
rotateY(*angle*)	定义沿着 Y 轴的 3D 旋转。	测试
rotateZ(*angle*)	定义沿着 Z 轴的 3D 旋转。	测试
skew(*x-angle*,*y-angle*)	定义沿着 X 和 Y 轴的 2D 倾斜转换。	测试
skewX(*angle*)	定义沿着 X 轴的 2D 倾斜转换。	测试
skewY(*angle*)	定义沿着 Y 轴的 2D 倾斜转换。	测试
perspective(*n*)	为 3D 转换元素定义透视视图。	测试

亲自试一试 - 实例

扔到桌子上面的图片

本例演示如何创建“宝丽来”图片，并旋转图片。

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
margin:30px;
background-color:#E9E9E9;
}

div.polaroid
{
width:294px;
```

```
padding:10px 10px 20px 10px;
border:1px solid #BFBFBF;
background-color:white;
/* Add box-shadow */
box-shadow:2px 2px 3px #aaaaaa;
}

div.rotate_left
{
float:left;
-ms-transform:rotate(7deg); /* IE 9 */
-moz-transform:rotate(7deg); /* Firefox */
-webkit-transform:rotate(7deg); /* Safari and Chrome */
-o-transform:rotate(7deg); /* Opera */
transform:rotate(7deg);
}

div.rotate_right
{
float:left;
-ms-transform:rotate(-8deg); /* IE 9 */
-moz-transform:rotate(-8deg); /* Firefox */
-webkit-transform:rotate(-8deg); /* Safari and Chrome */
-o-transform:rotate(-8deg); /* Opera */
transform:rotate(-8deg);
}
</style>
</head>
<body>

<div class="polaroid rotate_left">
<img src="/i/ballade_dream.jpg" alt="郁金香" width="284" height="213
<p class="caption">上海鲜花港的郁金香，花名：Ballade Dream。</p>
</div>

<div class="polaroid rotate_right">
<img src="/i/china_pavilion.jpg" alt="世博中国馆" width="284" height
<p class="caption">2010年上海世博会，中国馆。</p>
</div>

</body>
</html>
```

相关页面

CSS3 教程：CSS3 2D 转换

CSS3 教程：CSS3 3D 转换

CSS3 transform-origin 属性

实例

设置旋转元素的基点位置：

```
div
{
transform: rotate(45deg);
transform-origin:20% 40%;

-ms-transform: rotate(45deg);          /* IE 9 */
-ms-transform-origin:20% 40%;          /* IE 9 */

-webkit-transform: rotate(45deg);    /* Safari 和 Chrome */
-webkit-transform-origin:20% 40%;    /* Safari 和 Chrome */

-moz-transform: rotate(45deg);         /* Firefox */
-moz-transform-origin:20% 40%;         /* Firefox */

-o-transform: rotate(45deg);         /* Opera */
-o-transform-origin:20% 40%;         /* Opera */
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 支持 transform-origin 属性。

Internet Explorer 9 支持替代的 -ms-transform-origin 属性（仅适用于 2D 转换）。

Safari 和 Chrome 支持替代的 -webkit-transform-origin 属性（3D 和 2D 转换）。

Opera 只支持 2D 转换。

定义和用法

transform-origin 属性允许您改变被转换元素的位置。

2D 转换元素能够改变元素 x 和 y 轴。3D 转换元素还能改变其 Z 轴。

为了更好地理解 transform-origin 属性，请查看这个演示。

Safari/Chrome 用户：为了更好地理解 transform-origin 属性用于 3D 转换的情况，请查看这个演示。

注释：该属性必须与 transform 属性一同使用。

为了更好地理解 transform 属性，请查看这个演示。

默认值：	50% 50% 0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.transformOrigin="20% 40%"

语法

```
transform-origin: _x-axis_ _y-axis_ _z-axis_;
```

值	描述
x-axis	

定义视图被置于 X 轴的何处。可能的值：

- left
- center
- right
- *length*
- *%*

  | | y-axis |

定义视图被置于 Y 轴的何处。可能的值：

- top
- center
- bottom
- *length*
- *%*

  | | z-axis |

定义视图被置于 Z 轴的何处。可能的值：

- *length*

  |

相关页面

CSS3 教程：CSS3 2D 转换

CSS3 教程：CSS3 3D 转换

CSS3 transform-style 属性

实例

使被转换的子元素保留其 3D 转换：

```
div
{
transform: rotateY(60deg);
transform-style: preserve-3d;
-webkit-transform: rotateY(60deg);	/* Safari 和 Chrome */
-webkit-transform-style: preserve-3d;	/* Safari 和 Chrome */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Firefox 支持 transform-style 属性。

Chrome、Safari 和 Opera 支持替代的 -webkit-transform-style 属性。

定义和用法

transform-style 属性规定如何在 3D 空间中呈现被嵌套的元素。

注释：该属性必须与 transform 属性一同使用。

默认值：	flat
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.transformStyle="preserve-3d"

语法

```
transform-style: flat|preserve-3d;
```

值	描述
flat	子元素将不保留其 3D 位置。
preserve-3d	子元素将保留其 3D 位置。

CSS3 perspective 属性

实例

设置元素被查看位置的视图：

```
div
{
perspective: 500;
-webkit-perspective: 500; /* Safari 和 Chrome */
}
```

浏览器支持

目前浏览器都不支持 perspective 属性。

Chrome 和 Safari 支持替代的 -webkit-perspective 属性。

定义和用法

perspective 属性定义 3D 元素距视图的距离，以像素计。该属性允许您改变 3D 元素查看 3D 元素的视图。

当为元素定义 perspective 属性时，其子元素会获得透视效果，而不是元素本身。

注释：perspective 属性只影响 3D 转换元素。

提示：请与 perspective-origin 属性一同使用该属性，这样您就能够改变 3D 元素的底部位置。

默认值：	none
继承性：	yes
版本：	CSS3
JavaScript 语法：	*object*.style.perspective=500

语法

```
perspective: _number_|none;
```

值	描述
number	元素距离视图的距离，以像素计。
none	默认值。与 0 相同。不设置透视。

CSS3 perspective-origin 属性

实例

设置 3D 元素的基点位置：

```
div
{
perspective:150;
perspective-origin: 10% 10%;
-webkit-perspective:150;    /* Safari 和 Chrome */
-webkit-perspective-origin: 10% 10%;    /* Safari 和 Chrome */
}
```

浏览器支持

目前浏览器都不支持 perspective-origin 属性。

Chrome 和 Safari 支持替代的 -webkit-perspecitve-origin 属性。

定义和用法

perspective-origin 属性定义 3D 元素所基于的 X 轴和 Y 轴。该属性允许您改变 3D 元素的底部位置。

当为元素定义 perspective-origin 属性时，其子元素会获得透视效果，而不是元素本身。

注释：该属性必须与 perspective 属性一同使用，而且只影响 3D 转换元素。

默认值：	50% 50%
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.perspectiveOrigin="10% 10%"

语法

```
perspective-origin: _x-axis_ _y-axis_;
```

值	描述
x-axis	

定义该视图在 x 轴上的位置。默认值：50%。

可能的值：

- left
- center
- right
- *length*
- *%*

  | | *y-axis* |

定义该视图在 y 轴上的位置。默认值：50%。

可能的值：

- top
- center
- bottom
- *length*
- *%*

  |

CSS3 backface-visibility 属性

实例

隐藏被旋转的 div 元素的背面：

```
div
{
backface-visibility:hidden;
-webkit-backface-visibility:hidden;    /* Chrome 和 Safari */
-moz-backface-visibility:hidden;     /* Firefox */
-ms-backface-visibility:hidden;     /* Internet Explorer */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

只有 Internet Explorer 10+ 和 Firefox 支持 backface-visibility 属性。

Opera 15+、Safari 和 Chrome 支持替代的 -webkit-backface-visibility 属性。

定义和用法

backface-visibility 属性定义当元素不面向屏幕时是否可见。

如果在旋转元素不希望看到其背面时，该属性很有用。

默认值：	visible
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.backfaceVisibility="hidden"

语法

```
backface-visibility: visible|hidden;
```

值	描述
visible	背面是可见的。
hidden	背面是不可见的。

过渡属性（Transition）

属性	描述	CSS
transition	简写属性，用于在一个属性中设置四个过渡属性。	3
transition-property	规定应用过渡的 CSS 属性的名称。	3
transition-duration	定义过渡效果花费的时间。	3
transition-timing-function	规定过渡效果的时间曲线。	3
transition-delay	规定过渡效果何时开始。	3

CSS3 transition 属性

实例

把鼠标指针放到 div 元素上，其宽度会从 100px 逐渐变为 300px：

```
div
{
width:100px;
transition: width 2s;
-moz-transition: width 2s; /* Firefox 4 */
-webkit-transition: width 2s; /* Safari 和 Chrome */
-o-transition: width 2s; /* Opera */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 和 Chrome 支持 transition 属性。

Safari 支持替代的 -webkit-transition 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 transition 属性。

定义和用法

transition 属性是一个简写属性，用于设置四个过渡属性：

- transition-property
- transition-duration
- transition-timing-function
- transition-delay

注释：请始终设置 transition-duration 属性，否则时长为 0，就不会产生过渡效果。

默认值：	**all 0 ease 0**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.transition="width 2s"

语法

```
transition: _property_ _duration_ _timing-function_ _delay_;
```

值	描述
transition-property	规定设置过渡效果的 CSS 属性的名称。
transition-duration	规定完成过渡效果需要多少秒或毫秒。
transition-timing-function	规定速度效果的速度曲线。
transition-delay	定义过渡效果何时开始。

CSS3 transition-property 属性

实例

把鼠标指针放到 div 元素上，会产生带有平滑改变元素宽度的过渡效果：

```
div
{
transition-property:width;
-moz-transition-property: width; /* Firefox 4 */
-webkit-transition-property:width; /* Safari 和 Chrome */
-o-transition-property:width; /* Opera */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 和 Chrome 支持 transition-property 属性。

Safari 支持替代的 -webkit-transition-property 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 transition-property 属性。

定义和用法

transition-property 属性规定应用过渡效果的 CSS 属性的名称。（当指定的 CSS 属性改变时，过渡效果将开始）。

提示：过渡效果通常在用户将鼠标指针浮动到元素上时发生。

注释：请始终设置 transition-duration 属性，否则时长为 0，就不会产生过渡效果。

默认值：	**all**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.transitionProperty="width,height"

语法

```
transition-property: none|all|_property_;
```

值	描述
none	没有属性会获得过渡效果。
all	所有属性都将获得过渡效果。
property	定义应用过渡效果的 CSS 属性名称列表，列表以逗号分隔。

CSS3 transition-duration 属性

实例

让过渡效果持续 5 秒：

```
div
{
transition-duration: 5s;
-moz-transition-duration: 5s; /* Firefox 4 */
-webkit-transition-duration: 5s; /* Safari 和 Chrome */
-o-transition-duration: 5s; /* Opera */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 和 Chrome 支持 transition-duration 属性。

Safari 支持替代的 -webkit-transition-duration 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 transition-duration 属性。

定义和用法

transition-duration 属性规定完成过渡效果需要花费的时间（以秒或毫秒计）。

默认值：	**0**
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.transitionDuration="5s"
---	---

语法

```
transition-duration: _time_;
```

值	描述
time	

规定完成过渡效果需要花费的时间（以秒或毫秒计）。

默认值是 0，意味着不会有效果。

|

CSS3 transition-timing-function 属性

实例

以相同的速度从开始到结束的过渡效果：

```
div
{
transition-timing-function: linear;
-moz-transition-timing-function: linear; /* Firefox 4 */
-webkit-transition-timing-function: linear; /* Safari 和 Chrome */
-o-transition-timing-function: linear; /* Opera */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 和 Chrome 支持 transition-timing-function 属性。

Safari 支持替代的 -webkit-transition-timing-function 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 transition-timing-function 属性。

定义和用法

transition-timing-function 属性规定过渡效果的速度曲线。

该属性允许过渡效果随着时间来改变其速度。

默认值：	ease
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.transitionTimingFunction="linear"

语法

```
transition-timing-function: linear|ease|ease-in|ease-out|ease-in-ou
bezier(_n_,_n_,_n_,_n_);
```

值	描述
linear	规定以相同速度开始至结束的过渡效果（等于 cubic-bezier(0,0,1,1)）。
ease	规定慢速开始，然后变快，然后慢速结束的过渡效果（cubic-bezier(0.25,0.1,0.25,1)）。
ease-in	规定以慢速开始的过渡效果（等于 cubic-bezier(0.42,0,1,1)）。
ease-out	规定以慢速结束的过渡效果（等于 cubic-bezier(0,0,0.58,1)）。
ease-in-out	规定以慢速开始和结束的过渡效果（等于 cubic-bezier(0.42,0,0.58,1)）。
cubic-bezier(*n*,*n*,*n*,*n*)	在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。

提示：请在实例中测试不同的值，这样可以更好地理解它们的工作原理。

亲自试一试 - 实例

实例 1

为了更好地理解不同的函数值，请看下面带有五个不同值的五个不同的 div 元素：

```
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
/* Firefox 4: */
#div1 {-moz-transition-timing-function: linear;}
#div2 {-moz-transition-timing-function: ease;}
#div3 {-moz-transition-timing-function: ease-in;}
#div4 {-moz-transition-timing-function: ease-out;}
#div5 {-moz-transition-timing-function: ease-in-out;}
/* Safari and Chrome: */
#div1 {-webkit-transition-timing-function: linear;}
#div2 {-webkit-transition-timing-function: ease;}
#div3 {-webkit-transition-timing-function: ease-in;}
#div4 {-webkit-transition-timing-function: ease-out;}
#div5 {-webkit-transition-timing-function: ease-in-out;}
/* Opera: */
#div1 {-o-transition-timing-function: linear;}
#div2 {-o-transition-timing-function: ease;}
#div3 {-o-transition-timing-function: ease-in;}
#div4 {-o-transition-timing-function: ease-out;}
#div5 {-o-transition-timing-function: ease-in-out;}
```

实例 2

与上例相同，但通过 cubic-bezier 来规定速度曲线：

```
#div1 {transition-timing-function: cubic-bezier(0,0,1,1;}
#div2 {transition-timing-function: cubic-bezier(0.25,0.1,0.25,1);}
#div3 {transition-timing-function: cubic-bezier(0.42,0,1,1);}
#div4 {transition-timing-function: cubic-bezier(0,0,0.58,1);}
#div5 {transition-timing-function: cubic-bezier(0.42,0,0.58,1);}
/* Firefox 4: */
#div1 {-moz-transition-timing-function: cubic-bezier(0,0,0.25,1);}
#div2 {-moz-transition-timing-function: cubic-bezier(0.25,0.1,0.25,
#div3 {-moz-transition-timing-function: cubic-bezier(0.42,0,1,1);}
#div4 {-moz-transition-timing-function: cubic-bezier(0,0,0.58,1);}
#div5 {-moz-transition-timing-function: cubic-bezier(0.42,0,0.58,1
/* Safari and Chrome: */
#div1 {-webkit-transition-timing-function: cubic-bezier(0,0,1,1;}
#div2 {-webkit-transition-timing-function: cubic-bezier(0.25,0.1,0
#div3 {-webkit-transition-timing-function: cubic-bezier(0.42,0,1,1
#div4 {-webkit-transition-timing-function: cubic-bezier(0,0,0.58,1
#div5 {-webkit-transition-timing-function: cubic-bezier(0.42,0,0.5
/* Opera: */
#div1 {-o-transition-timing-function: cubic-bezier(0,0,1,1;}
#div2 {-o-transition-timing-function: cubic-bezier(0.25,0.1,0.25,1
#div3 {-o-transition-timing-function: cubic-bezier(0.42,0,1,1);}
#div4 {-o-transition-timing-function: cubic-bezier(0,0,0.58,1);}
#div5 {-o-transition-timing-function: cubic-bezier(0.42,0,0.58,1);
```

CSS3 transition-delay 属性

实例

在过渡效果开始前等待 2 秒：

```
div
{
transition-delay: 2s;
-moz-transition-delay: 2s; /* Firefox 4 */
-webkit-transition-delay: 2s; /* Safari 和 Chrome */
-o-transition-delay: 2s; /* Opera */
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer 10、Firefox、Opera 和 Chrome 支持 transition-delay 属性。

Safari 支持替代的 -webkit-transition-delay 属性。

注释：Internet Explorer 9 以及更早版本的浏览器不支持 transition-delay 属性。

定义和用法

transition-delay 属性规定过渡效果何时开始。

transition-delay 值以秒或毫秒计。

默认值：	0
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.transitionDelay="2s"
---	---

语法

```
transition-delay: _time_;
```

值	描述
time	规定在过渡效果开始之前需要等待的时间，以秒或毫秒计。

用户界面属性（User-interface）

属性	描述	CSS
appearance	允许您将元素设置为标准用户界面元素的外观	3
box-sizing	允许您以确切的方式定义适应某个区域的具体内容。	3
icon	为创作者提供使用图标化等价物来设置元素样式的能力。	3
nav-down	规定在使用 arrow-down 导航键时向何处导航。	3
nav-index	设置元素的 tab 键控制次序。	3
nav-left	规定在使用 arrow-left 导航键时向何处导航。	3
nav-right	规定在使用 arrow-right 导航键时向何处导航。	3
nav-up	规定在使用 arrow-up 导航键时向何处导航。	3
outline-offset	对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。	3
resize	规定是否可由用户对元素的尺寸进行调整。	3

CSS3 appearance 属性

实例

使 div 元素看上去像一个按钮：

```
div
{
appearance:button;
-moz-appearance:button; /* Firefox */
-webkit-appearance:button; /* Safari 和 Chrome */
}
```

页面底部有更多实例。

浏览器支持

所有主流浏览器都不支持 appearance 属性。

Firefox 支持替代的 -moz-appearance 属性。

Safari 和 Chrome 支持替代的 -webkit-appearance 属性。

定义和用法

appearance 属性允许您使元素看上去像标准的用户界面元素。

默认值：	normal
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.appearance="button"

语法

```
appearance: normal|icon|window|button|menu|field;
```

值	描述
normal	将元素呈现为常规元素。
icon	将元素呈现为图标（小图片）。
window	将元素呈现为视口。
button	将元素呈现为按钮。
menu	将元素呈现为一套供用户选择的选项。
field	将元素呈现为输入字段。

CSS3 box-sizing 属性

实例

规定两个并排的带边框的框：

```
div
{
box-sizing:border-box;
-moz-box-sizing:border-box; /* Firefox */
-webkit-box-sizing:border-box; /* Safari */
width:50%;
float:left;
}
```

页面底部有更多实例。

浏览器支持

IE	Firefox	Chrome	Safari	Opera

Internet Explorer、Opera 以及 Chrome 支持 box-sizing 属性。

Firefox 支持替代的 -moz-box-sizing 属性。

定义和用法

box-sizing 属性允许您以特定的方式定义匹配某个区域的特定元素。

例如，假如您需要并排放置两个带边框的框，可通过将 box-sizing 设置为 "border-box"。这可令浏览器呈现出带有指定宽度和高度的框，并把边框和内边距放入框中。

默认值：	content-box
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.boxSizing="border-box"

语法

```
box-sizing: content-box|border-box|inherit;
```

值	描述
content-box	

这是由 CSS2.1 规定的宽度高度行为。

宽度和高度分别应用到元素的内容框。

在宽度和高度之外绘制元素的内边距和边框。

| | border-box |

为元素设定的宽度和高度决定了元素的边框盒。

就是说，为元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。

通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。

| | inherit | 规定应从父元素继承 box-sizing 属性的值。 |

相关页面

CSS3 教程：CSS3 用户界面

CSS3 icon 属性

实例

将图像元素设置为图标化的等价物：

```
img
{
content:icon;
icon:url(imgicon.png);
}
```

浏览器支持

IE	Firefox	Chrome	Safari	Opera

目前没有浏览器支持 icon 属性。

定义和用法

icon 属性为创作者提供了将元素设置为图标等价物的能力。

注释：除非 "content" 属性的值被设置为 "icon"，否则元素的图标不会被使用。

默认值：	auto
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.icon="url(image.png)"

语法

```
icon: auto|_URL_|inherit;
```

值	描述
auto	使用由浏览器提供的默认通用图标。
URL	引用列表中的一个或多个图标，列表用逗号分隔。
inherit	规定应从元素继承 icon 属性的值。

CSS3 nav-down 属性

实例

规定在使用方向键时向何处导航：

```
button#b1
{
top:20%;left:25%;
nav-index:1;
nav-right:#b2;nav-left:#b4;
nav-down:#b2;nav-up:#b4;
}

button#b2
{
top:40%;left:50%;
nav-index:2;
nav-right:#b3;nav-left:#b1;
nav-down:#b3;nav-up:#b1;
}

button#b3
{
top:70%;left:25%;
nav-index:3;
nav-right:#b4;nav-left:#b2;
nav-down:#b4;nav-up:#b2;
}

button#b4
{
top:40%;left:0%;
nav-index:4;
nav-right:#b1;nav-left:#b3;
nav-down:#b1;nav-up:#b3;
}
```

浏览器支持

目前只有 Opera 支持 nav-down 属性。

定义和用法

nav-down 属性规定当使用 nav-down 导航键时，向何处进行导航。

默认值：	**auto**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.navDown="#div2"

语法

```
nav-down: auto|_id_|_target-name_|inherit;
```

值	描述
auto	浏览器决定导航到哪个元素。
id	规定被导航元素的 id。
target-name	规定被导航的目标框架。
inherit	规定应从父元素继承 nav-down 属性的值。

CSS3 nav-index 属性

实例

规定在使用方向键时向何处导航：

```
button#b1
{
top:20%;left:25%;
nav-index:1;
nav-right:#b2;nav-left:#b4;
nav-down:#b2;nav-up:#b4;
}

button#b2
{
top:40%;left:50%;
nav-index:2;
nav-right:#b3;nav-left:#b1;
nav-down:#b3;nav-up:#b1;
}

button#b3
{
top:70%;left:25%;
nav-index:3;
nav-right:#b4;nav-left:#b2;
nav-down:#b4;nav-up:#b2;
}

button#b4
{
top:40%;left:0%;
nav-index:4;
nav-right:#b1;nav-left:#b3;
nav-down:#b1;nav-up:#b3;
}
```

浏览器支持

目前只有 Opera 支持 nav-index 属性。

定义和用法

nav-index 属性规定元素的连续导航次序 ("tabbing order")。

默认值：	**auto**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.navIndex=2

语法

```
nav-index: auto|_number_|inherit;
```

值	描述
auto	由浏览器分配元素的导航键控制次序。
number	指示元素的导航键控制次序。1 代表第一个。
inherit	规定应从父元素继承 nav-index 属性的值。

CSS3 nav-left 属性

实例

规定在使用方向键时向何处导航：

```
button#b1
{
top:20%;left:25%;
nav-index:1;
nav-right:#b2;nav-left:#b4;
nav-down:#b2;nav-up:#b4;
}

button#b2
{
top:40%;left:50%;
nav-index:2;
nav-right:#b3;nav-left:#b1;
nav-down:#b3;nav-up:#b1;
}

button#b3
{
top:70%;left:25%;
nav-index:3;
nav-right:#b4;nav-left:#b2;
nav-down:#b4;nav-up:#b2;
}

button#b4
{
top:40%;left:0%;
nav-index:4;
nav-right:#b1;nav-left:#b3;
nav-down:#b1;nav-up:#b3;
}
```

浏览器支持

目前只有 Opera 支持 nav-left 属性。

定义和用法

nav-left 属性规定当使用 nav-left 导航键时，向何处进行导航。

默认值：	**auto**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.navLeft="#div2"

语法

```
nav-left: auto|_id_|_target-name_|inherit;
```

值	描述
auto	浏览器决定导航到哪个元素。
id	规定被导航元素的 id。
target-name	规定被导航的目标框架。
inherit	规定应从父元素继承 nav-left 属性的值。

CSS3 nav-right 属性

实例

规定在使用方向键时向何处导航：

```
button#b1
{
top:20%;left:25%;
nav-index:1;
nav-right:#b2;nav-left:#b4;
nav-down:#b2;nav-up:#b4;
}

button#b2
{
top:40%;left:50%;
nav-index:2;
nav-right:#b3;nav-left:#b1;
nav-down:#b3;nav-up:#b1;
}

button#b3
{
top:70%;left:25%;
nav-index:3;
nav-right:#b4;nav-left:#b2;
nav-down:#b4;nav-up:#b2;
}

button#b4
{
top:40%;left:0%;
nav-index:4;
nav-right:#b1;nav-left:#b3;
nav-down:#b1;nav-up:#b3;
}
```

浏览器支持

目前只有 Opera 支持 nav-right 属性。

定义和用法

nav-right 属性规定当使用 nav-right 导航键时，向何处进行导航。

默认值：	**auto**
继承性：	no
---	---
版本：	CSS3
---	---
JavaScript 语法：	*object*.style.navRight="#div2"
---	---

语法

```
nav-right: auto|_id_|_target-name_|inherit;
```

值	描述
auto	浏览器决定导航到哪个元素。
id	规定被导航元素的 id。
target-name	规定被导航的目标框架。
inherit	规定应从父元素继承 nav-right 属性的值。

CSS3 nav-up 属性

实例

规定在使用方向键时向何处导航：

```
button#b1
{
top:20%;left:25%;
nav-index:1;
nav-right:#b2;nav-left:#b4;
nav-down:#b2;nav-up:#b4;
}

button#b2
{
top:40%;left:50%;
nav-index:2;
nav-right:#b3;nav-left:#b1;
nav-down:#b3;nav-up:#b1;
}

button#b3
{
top:70%;left:25%;
nav-index:3;
nav-right:#b4;nav-left:#b2;
nav-down:#b4;nav-up:#b2;
}

button#b4
{
top:40%;left:0%;
nav-index:4;
nav-right:#b1;nav-left:#b3;
nav-down:#b1;nav-up:#b3;
}
```

浏览器支持

目前只有 Opera 支持 nav-up 属性。

定义和用法

nav-up 属性规定当使用 nav-up 导航键时，向何处进行导航。

默认值：	**auto**
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.navUp="#div2"

语法

```
nav-up: auto|_id_|_target-name_|inherit;
```

值	描述
auto	浏览器决定导航到哪个元素。
id	规定被导航元素的 id。
target-name	规定被导航的目标框架。
inherit	规定应从父元素继承 nav-up 属性的值。

CSS3 outline-offset 属性

实例

规定边框边缘之外 15 像素处的轮廓：

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

浏览器支持

所有主流浏览器都支持 outline-offset 属性，除了 Internet Explorer。

定义和用法

outline-offset 属性对轮廓进行偏移，并在边框边缘进行绘制。

轮廓在两方面与边框不同：

- 轮廓不占用空间
- 轮廓可能是非矩形

默认值：	0
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.outlineOffset="15px"

语法

```
outline-offset: _length_|inherit;
```

值	描述
length	轮廓与边框边缘的距离。
inherit	规定应从父元素继承 outline-offset 属性的值。

CSS3 resize 属性

实例

规定可以由用户调整 div 元素的大小：

```
div
{
resize:both;
overflow:auto;
}
```

浏览器支持

Firefox 4+、Chrome 以及 Safari 不支持 resize。

定义和用法

resize 属性规定是否可由用户调整元素的尺寸。

注释：如果希望此属性生效，需要设置元素的 overflow 属性，值可以是 auto、hidden 或 scroll。

默认值：	none
继承性：	no
版本：	CSS3
JavaScript 语法：	*object*.style.resize="both"

语法

```
resize: none|both|horizontal|vertical;
```

值	描述
none	用户无法调整元素的尺寸。
both	用户可调整元素的高度和宽度。
horizontal	用户可调整元素的宽度。
vertical	用户可调整元素的高度。

相关页面

CSS3 教程：CSS3 用户界面

CSS 选择器参考手册

我们会定期对 W3School 的 CSS 参考手册进行浏览器测试。

CSS3 选择器

在 CSS 中，选择器是一种模式，用于选择需要添加样式的元素。

"CSS" 列指示该属性是在哪个 CSS 版本中定义的。（CSS1、CSS2 还是 CSS3。）

选择器	例子	例子描述	CSS
.class	.intro	选择 class="intro" 的所有元素。	1
#id	#firstname	选择 id="firstname" 的所有元素。	1
*	*	选择所有元素。	2
element	p	选择所有 <p> 元素。	1
element,element	div,p	选择所有 <div> 元素和所有 <p> 元素。	1
element element	div p	选择 <div> 元素内部的所有 <p> 元素。	1
element>element	div>p	选择父元素为 <div> 元素的所有 <p> 元素。	2
element+element	div+p	选择紧接在 <div> 元素之后的所有 <p> 元素。	2
[*attribute*]	[target]	选择带有 target 属性所有元素。	2
[*attribute=value*]	[target=_blank]	选择 target="_blank" 的所有元素。	2
[*attribute~=value*]	[title~=flower]	选择 title 属性包含单词 "flower" 的所有元素。	2
[*attribute*|=*value*]	[lang|=en]	选择 lang 属性值以 "en" 开头的所有元素。	2
:link	a:link	选择所有未被访问的链接。	1

:active	a:active	选择活动链接。	1
:hover	a:hover	选择鼠标指针位于其上的链接。	1
:focus	input:focus	选择获得焦点的 input 元素。	2
:first-letter	p:first-letter	选择每个 <p> 元素的首字母。	1
:first-line	p:first-line	选择每个 <p> 元素的首行。	1
:first-child	p:first-child	选择属于父元素的第一个子元素的每个 <p> 元素。	2
:before	p:before	在每个 <p> 元素的内容之前插入内容。	2
:after	p:after	在每个 <p> 元素的内容之后插入内容。	2
:lang(*language*)	p:lang(it)	选择带有以 "it" 开头的 lang 属性值的每个 <p> 元素。	2
element1~element2	p~ul	选择前面有 <p> 元素的每个 元素。	3
[*attribute^=value*]	a[src^="https"]	选择其 src 属性值以 "https" 开头的每个 <a> 元素。	3
[*attribute$=value*]	a[src$=".pdf"]	选择其 src 属性以 ".pdf" 结尾的所有 <a> 元素。	3
[*attribute*=value*]	a[src*="abc"]	选择其 src 属性中包含 "abc" 子串的每个 <a> 元素。	3
:first-of-type	p:first-of-type	选择属于其父元素的首个 <p> 元素的每个 <p> 元素。	3
:last-of-type	p:last-of-type	选择属于其父元素的最后 <p> 元素的每个 <p> 元素。	3
:only-of-type	p:only-of-type	选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。	3
:only-child	p:only-child	选择属于其父元素的唯一子元素的每个 <p> 元素。	3
:nth-child(*n*)	p:nth-child(2)	选择属于其父元素的第二个子元素的每个 <p> 元素。	3
:nth-last-child(*n*)	p:nth-last-child(2)	同上，从最后一个子元素开始计数。	3

:nth-of-type(*n*)	type(2)	<p> 元素的每个 <p> 元素。	3
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	同上，但是从最后一个子元素开始计数。	3
:last-child	p:last-child	选择属于其父元素最后一个子元素每个 <p> 元素。	3
:root	:root	选择文档的根元素。	3
:empty	p:empty	选择没有子元素的每个 <p> 元素（包括文本节点）。	3
:target	#news:target	选择当前活动的 #news 元素。	3
:enabled	input:enabled	选择每个启用的 <input> 元素。	3
:disabled	input:disabled	选择每个禁用的 <input> 元素	3
:checked	input:checked	选择每个被选中的 <input> 元素。	3
:not(*selector*)	:not(p)	选择非 <p> 元素的每个元素。	3
::selection	::selection	选择被用户选取的元素部分。	3

CSS 听觉参考手册

听觉样式表

听觉样式表可把语音合成与音响效果相组合，使用户可以听到信息，而无需进行阅读。

听觉呈现可用于：

- 视觉能力低弱的人士
- 帮助用户学习阅读
- 帮助有阅读障碍的用户
- 家庭娱乐
- 在汽车中使用

听觉呈现通常会把文档转化为纯文本，然后传给屏幕阅读器（可读出屏幕上所有字符的一种程序）。

听觉样式表的一个例子：

```
h1, h2, h3, h4
{
voice-family: male;
richness: 80;
cue-before: url("beep.au")
}
```

上面的例子可以让语音合成器演奏一段声音，然后用男性的声音读出标题。

CSS2 听觉参考

W3C："W3C" 列的数字显示出属性由哪个 CSS 标准定义（CSS1 还是 CSS2）。

属性	描述	值	W3C
azimuth	Sets where the sound/voices should come from (horizontally)	angle left-side far-left left center-left center center-right right far-right right-side behind leftwards rightwards	2
	A shorthand property for		

cue	setting the cue-before and cue-after properties in one declaration	`cue-before` `cue-after`	2
cue-after	Specifies a sound to be played after speaking an element's content to delimit it from other	`none` `url`	2
cue-before	Specifies a sound to be played before speaking an element's content to delimit it from other	`none` `url`	2
elevation	Sets where the sound/voices should come from (vertically)	`angle` `belo` `level` `above` `higher` `lower`	2
pause	A shorthand property for setting the pause-before and pause-after properties in one declaration	`pause-before` `pause-after`	2
pause-after	Specifies a pause after speaking an element's content	`time` `%`	2
pause-before	Specifies a pause before speaking an element's content	`time` `%`	2
pitch	Specifies the speaking voice	`frequency` `x-low` `low` `medium` `high` `x-high`	2
pitch-range	Specifies the variation in the speaking voice. (Monotone voice or animated voice?)	`number`	2
play-during	Specifies a sound to be played while speaking an element's content	`auto` `none` `url` `mix` `repeat`	2
richness	Specifies the richness in the speaking voice. (Rich voice or thin voice?)	`number`	2
speak	Specifies whether content will render	`normal` `none` `spell-out`	2

	aurally	`spell-out`	
speak-header	Specifies how to handle table headers. Should the headers be spoken before every cell, or only before a cell with a different header than the previous cell	`always` `once`	2
speak-numeral	Specifies how to speak numbers	`digits` `continuous`	2
speak-punctuation	Specifies how to speak punctuation characters	`none` `code`	2
speech-rate	Specifies the speed of the speaking	`number` `x-slow` `slow` `medium` `fast` `x-fast` `faster` `slower`	2
stress	Specifies the "stress" in the speaking voice	`number`	2
voice-family	A prioritized list of voice family names that contain specific voices	`specific-voice` `generic-voice`	2
volume	Specifies the volume of the speaking	`number` `%` `silent` `x-soft` `soft` `medium` `loud` `x-loud`	2

CSS 网络安全字体组合

常用的字体组合

font-family 属性应该使用若干种字体名称作为回退系统，以确保浏览器/操作系统之间的最大兼容性。如果浏览器不支持第一个字体，则会尝试下一个。

请以您喜欢的字体开始，并以通用字体系列结束，以便使浏览器在通用系统中挑选相似的字体，如果没有其他字体可用的话：

实例

```
p{font-family:'Times New Roman', Times, serif}
```

下面是最常用的字体组合，根据通用系统进行汇总：

Serif 字体

font-family	示例文本
Georgia, serif	**This is a heading** This is a paragraph
'Palatino Linotype', 'Book Antiqua', Palatino, serif	**This is a heading** This is a paragraph
'Times New Roman', Times, serif	**This is a heading** This is a paragraph

---

Sans-Serif 字体

font-family	示例文本
Arial, Helvetica, sans-serif	**This is a heading** This is a paragraph
'Arial Black', Gadget, sans-serif	**This is a heading** **This is a paragraph**
'Comic Sans MS', cursive, sans-serif	**This is a heading** This is a paragraph
Impact, Charcoal, sans-serif	**This is a heading** **This is a paragraph**
'Lucida Sans Unicode', 'Lucida Grande', sans-serif	**This is a heading** This is a paragraph
Tahoma, Geneva, sans-serif	**This is a heading** This is a paragraph
'Trebuchet MS', Helvetica, sans-serif	**This is a heading** This is a paragraph
Verdana, Geneva, sans-serif	**This is a heading** This is a paragraph

---

Monospace 字体

font-family	示例文本
'Courier New', Courier, monospace	**This is a heading** This is a paragraph
'Lucida Console', Monaco, monospace	**This is a heading** This is a paragraph

CSS 单位

尺寸

单位	描述
%	百分比
in	英寸
cm	厘米
mm	毫米
em	1em 等于当前的字体尺寸。2em 等于当前字体尺寸的两倍。例如，如果某元素以 12pt 显示，那么 2em 是24pt。在 CSS 中，em 是非常有用的单位，因为它可以自动适应用户所使用的字体。
ex	一个 ex 是一个字体的 x-height。 (x-height 通常是字体尺寸的一半。)
pt	磅 (1 pt 等于 1/72 英寸)
pc	12 点活字 (1 pc 等于 12 点)
px	像素 (计算机屏幕上的一个点)

颜色

单位	描述
(颜色名)	颜色名称 (比如 red)
rgb(x,x,x)	RGB 值 (比如 rgb(255,0,0))
rgb(x%, x%, x%)	RGB 百分比值 (比如 rgb(100%,0%,0%))
#rrggbb	十六进制数 (比如 #ff0000)

CSS 颜色

颜色是通过对红、绿和蓝光的组合来显示的。

颜色值

CSS 颜色使用组合了红绿蓝颜色值 (RGB) 的十六进制 (hex) 表示法进行定义。对光源进行设置的最低值可以是 0（十六进制 00）。最高值是 255（十六进制 FF）。

十六进制值使用三个双位数来编写，并以 # 符号开头。

颜色	颜色 HEX	颜色 RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

1600 万种不同的颜色

从 0 到 255 种红绿蓝值能够组合出总共超过一千六百万种不同的颜色（根据 256 x 256 x 256 计算）。

大多数现代的显示器都能显示出至少 16384 种不同的颜色。

如果您查看下面的表格，您将看到红光从 0 到 255 变化后的结果，此时绿光和蓝光为零。

如需查看红光由 0 向 255 变化的完整颜色混合器列表，请点击下面的十六进制值或 rgb 值。

Red Light	HEX	RGB
	#000000	rgb(0,0,0)

	#080000	rgb(8,0,0)
	#100000	rgb(16,0,0)
	#180000	rgb(24,0,0)
	#200000	rgb(32,0,0)
	#280000	rgb(40,0,0)
	#300000	rgb(48,0,0)
	#380000	rgb(56,0,0)
	#400000	rgb(64,0,0)
	#480000	rgb(72,0,0)
	#500000	rgb(80,0,0)
	#580000	rgb(88,0,0)
	#600000	rgb(96,0,0)
	#680000	rgb(104,0,0)
	#700000	rgb(112,0,0)
	#780000	rgb(120,0,0)
	#800000	rgb(128,0,0)
	#880000	rgb(136,0,0)
	#900000	rgb(144,0,0)
	#980000	rgb(152,0,0)
	#A00000	rgb(160,0,0)
	#A80000	rgb(168,0,0)
	#B00000	rgb(176,0,0)
	#B80000	rgb(184,0,0)
	#C00000	rgb(192,0,0)
	#C80000	rgb(200,0,0)
	#D00000	rgb(208,0,0)
	#D80000	rgb(216,0,0)
	#E00000	rgb(224,0,0)
	#E80000	rgb(232,0,0)
	#F00000	rgb(240,0,0)

	#F80000	rgb(248,0,0)
	#FF0000	rgb(255,0,0)

灰阶

灰色使用所有光源的等量的光线来显示。为了让您更方便地选择正确的灰色，我们已经为您编辑了一个灰色列表：

灰阶	HEX	RGB
	#000000	rgb(0,0,0)
	#080808	rgb(8,8,8)
	#101010	rgb(16,16,16)
	#181818	rgb(24,24,24)
	#202020	rgb(32,32,32)
	#282828	rgb(40,40,40)
	#303030	rgb(48,48,48)
	#383838	rgb(56,56,56)
	#404040	rgb(64,64,64)
	#484848	rgb(72,72,72)
	#505050	rgb(80,80,80)
	#585858	rgb(88,88,88)
	#606060	rgb(96,96,96)
	#686868	rgb(104,104,104)
	#707070	rgb(112,112,112)
	#787878	rgb(120,120,120)
	#808080	rgb(128,128,128)
	#888888	rgb(136,136,136)
	#909090	rgb(144,144,144)
	#989898	rgb(152,152,152)
	#A0A0A0	rgb(160,160,160)
	#A8A8A8	rgb(168,168,168)

	#B0B0B0	rgb(176,176,176)
	#B8B8B8	rgb(184,184,184)
	#C0C0C0	rgb(192,192,192)
	#C8C8C8	rgb(200,200,200)
	#D0D0D0	rgb(208,208,208)
	#D8D8D8	rgb(216,216,216)
	#E0E0E0	rgb(224,224,224)
	#E8E8E8	rgb(232,232,232)
	#F0F0F0	rgb(240,240,240)
	#F8F8F8	rgb(248,248,248)
	#FFFFFF	rgb(255,255,255)

网络安全色？

多年前，当电脑只支持最多 256 种颜色时，216 种“网络安全色”列表被定义为 Web 标准，并保留了 40 种固定的系统颜色。

现在，这些都不重要了，因为大多数电脑都能显示数百万种颜色，但是选择权依然在您手里。

这个 216 跨浏览器调色板被创建的目的是确保当计算机运行 256 色调色板时能够正确地显示颜色：

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF

33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

CSS 合法颜色值

CSS 颜色

可以用以下方法来规定 CSS 中的颜色：

- 十六进制色
- RGB 颜色
- RGBA 颜色
- HSL 颜色
- HSLA 颜色
- 预定义/跨浏览器颜色名

十六进制颜色

所有浏览器都支持十六进制颜色值。

十六进制颜色是这样规定的：#RRGGBB，其中的 RR（红色）、GG（绿色）、BB（蓝色）十六进制整数规定了颜色的成分。所有值必须介于 0 与 FF 之间。

举例说，#0000ff 值显示为蓝色，这是因为蓝色成分被设置为最高值（ff），而其他成分被设置为 0。

实例

```
p
{
background-color:#0000ff;
}
```

RGB 颜色

所有浏览器都支持 RGB 颜色值。

RGB 颜色值是这样规定的：rgb(red, green, blue)。每个参数 (red、green 以及 blue) 定义颜色的强度，可以是介于 0 与 255 之间的整数，或者是百分比值（从 0% 到 100%）。

举例说，rgb(0,0,255) 值显示为蓝色，这是因为 blue 参数被设置为最高值（255），而其他被设置为 0。

同样地，下面的值定义了相同的颜色：rgb(0,0,255) 和 rgb(0%,0%,100%)。

实例

```
p
{
background-color:rgb(255,0,0);
}
```

RGBA 颜色

RGBA 颜色值得到以下浏览器的支持：IE9+、Firefox 3+、Chrome、Safari 以及 Opera 10+。

RGBA 颜色值是 RGB 颜色值的扩展，带有一个 alpha 通道 - 它规定了对象的不透明度。

RGBA 颜色值是这样规定的：rgba(red, green, blue, alpha)。alpha 参数是介于 0.0（完全透明）与 1.0（完全不透明）的数字。

实例

```
p
{
background-color:rgba(255,0,0,0.5);
}
```

HSL 颜色

HSL 颜色值得到以下浏览器的支持：IE9+、Firefox、Chrome、Safari 以及 Opera 10+。

HSL 指的是 hue（色调）、saturation（饱和度）、lightness（亮度） - 表示颜色柱面坐标表示法。

HSL 颜色值是这样规定的：hsl(hue, saturation, lightness)。

Hue 是色盘上的度数（从 0 到 360） - 0 (或 360) 是红色，120 是绿色，240 是蓝色。Saturation 是百分比值；0% 意味着灰色，而 100% 是全彩。Lightness 同样是百分比值；0% 是黑色，100% 是白色。

实例

```
p
{
background-color:hsl(120,65%,75%);
}
```

HSLA 颜色

HSLA 颜色值得到以下浏览器的支持：IE9+、Firefox 3+、Chrome、Safari 以及 Opera 10+。

HSLA 颜色值是 HSL 颜色值的扩展，带有一个 alpha 通道 - 它规定了对象的不透明度。

HSLA 颜色值是这样规定的：hsla(hue, saturation, lightness, alpha)，其中的 alpha 参数定义不透明度。alpha 参数是介于 0.0（完全透明）与 1.0（完全不透明）的数字。

实例

```
p
{
background-color:hsla(120,65%,75%,0.3);
}
```

CSS 颜色名

CSS 颜色名

所有浏览器都支持的颜色名。

HTML 和 CSS 颜色规范中定义了 147 中颜色名（17 种标准颜色加 130 种其他颜色）。下面的表格中列出了所有这些颜色，以及它们的十六进制值。

提示：17 种标准色是 aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow。

点击颜色名（或十六进制值）来查看这种颜色作为背景色搭配不同的文本颜色：

以颜色名排序

根据十六进制值排序的相同列表

颜色名	十六进制颜色值	颜色
AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	
Aqua	#00FFFF	
Aquamarine	#7FFFD4	
Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	

Chocolate	#D2691E
Coral	#FF7F50
CornflowerBlue	#6495ED
Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
Darkorange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray	#696969
DodgerBlue	#1E90FF
Feldspar	#D19275
FireBrick	#B22222
FloralWhite	#FFFAF0

ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC
GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateBlue	#8470FF

LightSlateGray	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0
Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6

PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE
PaleVioletRed	#D87093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8

Tomato	#FF6347
Turquoise	#40E0D0
Violet	#EE82EE
VioletRed	#D02090
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00
YellowGreen	#9ACD32

CSS 颜色十六进制值

根据十六进制值排序

所有浏览器都支持的颜色名

根据颜色名排序的相同列表

颜色名	十六进制颜色值	颜色
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	
DarkGreen	#006400	
Green	#008000	
Teal	#008080	
DarkCyan	#008B8B	
DeepSkyBlue	#00BFFF	
DarkTurquoise	#00CED1	
MediumSpringGreen	#00FA9A	
Lime	#00FF00	
SpringGreen	#00FF7F	
Aqua	#00FFFF	
Cyan	#00FFFF	
MidnightBlue	#191970	
DodgerBlue	#1E90FF	
LightSeaGreen	#20B2AA	
ForestGreen	#228B22	
SeaGreen	#2E8B57	
DarkSlateGray	#2F4F4F	

LimeGreen	#32CD32
MediumSeaGreen	#3CB371
Turquoise	#40E0D0
RoyalBlue	#4169E1
SteelBlue	#4682B4
DarkSlateBlue	#483D8B
MediumTurquoise	#48D1CC
Indigo	#4B0082
DarkOliveGreen	#556B2F
CadetBlue	#5F9EA0
CornflowerBlue	#6495ED
MediumAquaMarine	#66CDAA
DimGray	#696969
DimGrey	#696969
SlateBlue	#6A5ACD
OliveDrab	#6B8E23
SlateGray	#708090
LightSlateGray	#778899
MediumSlateBlue	#7B68EE
LawnGreen	#7CFC00
Chartreuse	#7FFF00
Aquamarine	#7FFFD4
Maroon	#800000
Purple	#800080
Olive	#808000
Gray	#808080
SkyBlue	#87CEEB
LightSkyBlue	#87CEFA
BlueViolet	#8A2BE2
DarkRed	#8B0000

DarkMagenta	#8B008B
SaddleBrown	#8B4513
DarkSeaGreen	#8FBC8F
LightGreen	#90EE90
MediumPurple	#9370DB
DarkViolet	#9400D3
PaleGreen	#98FB98
DarkOrchid	#9932CC
YellowGreen	#9ACD32
Sienna	#A0522D
Brown	#A52A2A
DarkGray	#A9A9A9
LightBlue	#ADD8E6
GreenYellow	#ADFF2F
PaleTurquoise	#AFEEEE
LightSteelBlue	#B0C4DE
PowderBlue	#B0E0E6
FireBrick	#B22222
DarkGoldenRod	#B8860B
MediumOrchid	#BA55D3
RosyBrown	#BC8F8F
DarkKhaki	#BDB76B
Silver	#C0C0C0
MediumVioletRed	#C71585
IndianRed	#CD5C5C
Peru	#CD853F
Chocolate	#D2691E
Tan	#D2B48C
LightGray	#D3D3D3
Thistle	#D8BFD8

Orchid	#DA70D6
GoldenRod	#DAA520
PaleVioletRed	#DB7093
Crimson	#DC143C
Gainsboro	#DCDCDC
Plum	#DDA0DD
BurlyWood	#DEB887
LightCyan	#E0FFFF
Lavender	#E6E6FA
DarkSalmon	#E9967A
Violet	#EE82EE
PaleGoldenRod	#EEE8AA
LightCoral	#F08080
Khaki	#F0E68C
AliceBlue	#F0F8FF
HoneyDew	#F0FFF0
Azure	#F0FFFF
SandyBrown	#F4A460
Wheat	#F5DEB3
Beige	#F5F5DC
WhiteSmoke	#F5F5F5
MintCream	#F5FFFA
GhostWhite	#F8F8FF
Salmon	#FA8072
AntiqueWhite	#FAEBD7
Linen	#FAF0E6
LightGoldenRodYellow	#FAFAD2
OldLace	#FDF5E6
Red	#FF0000

Fuchsia	#FF00FF
Magenta	#FF00FF
DeepPink	#FF1493
OrangeRed	#FF4500
Tomato	#FF6347
HotPink	#FF69B4
Coral	#FF7F50
Darkorange	#FF8C00
LightSalmon	#FFA07A
Orange	#FFA500
LightPink	#FFB6C1
Pink	#FFC0CB
Gold	#FFD700
PeachPuff	#FFDAB9
NavajoWhite	#FFDEAD
Moccasin	#FFE4B5
Bisque	#FFE4C4
MistyRose	#FFE4E1
BlanchedAlmond	#FFEBCD
PapayaWhip	#FFEFD5
LavenderBlush	#FFF0F5
SeaShell	#FFF5EE
Cornsilk	#FFF8DC
LemonChiffon	#FFFACD
FloralWhite	#FFFAF0
Snow	#FFFAFA
Yellow	#FFFF00
LightYellow	#FFFFE0
Ivory	#FFFFF0
White	#FFFFFF

Firebug 教程

Firebug 教程

什么是 Firebug？

Firebug 是一个开源的web开发工具。

在本章教程我们将讨论以下内容：

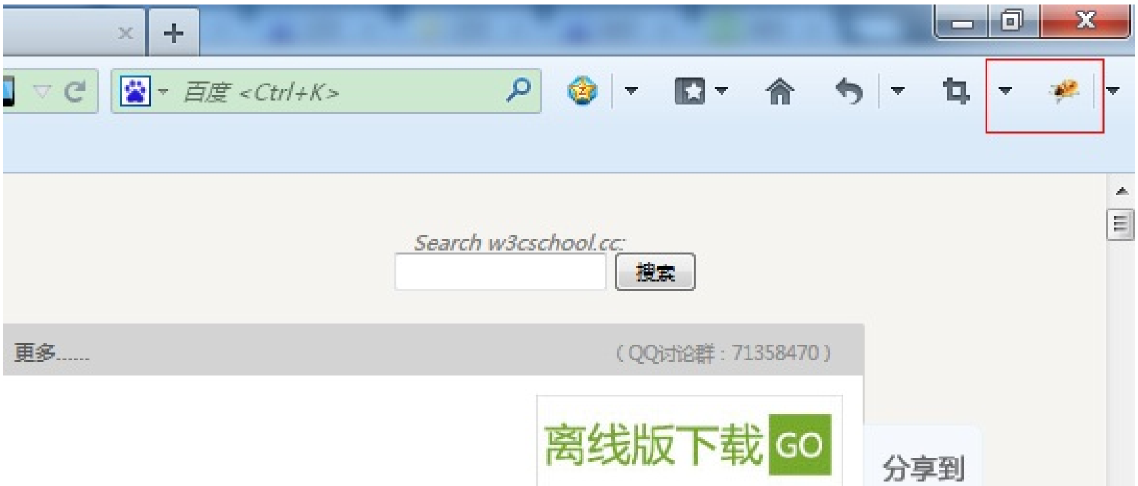
- 如何安装 Firebug。
- 使用 Firebug 检查和编辑 HTML。
- 使用 Firebug 检查和编辑 CSS。
- 使用 Firebug 调试 JavaScript。
- 在 Firebug 中动态执行 JavaScript。
- 使用 Firebug 记录 Javascript日志。
- 使用 Firebug 监控网络。

安装 Firebug

Firebug下载地址： https://addons.mozilla.org/en-US/firefox/addon/1843/

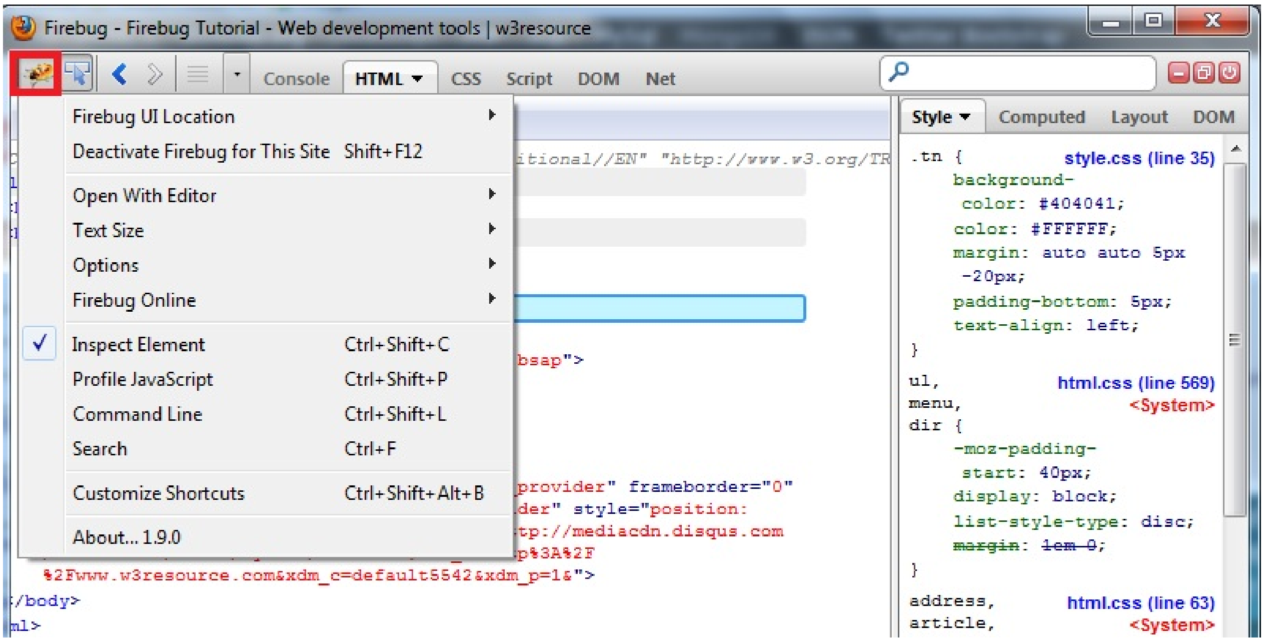
使用Firefox浏览器访问以上下载地址，打开页面后，点击下载按钮后，会有个弹窗页面，提示安装的目录，安装后重启你的Firefox浏览器。

重启后就可以在Firefox浏览器中看到Firebug的图标。点击Firebug图标(位于Firefox浏览器右上角)即可激活Firebug插件。

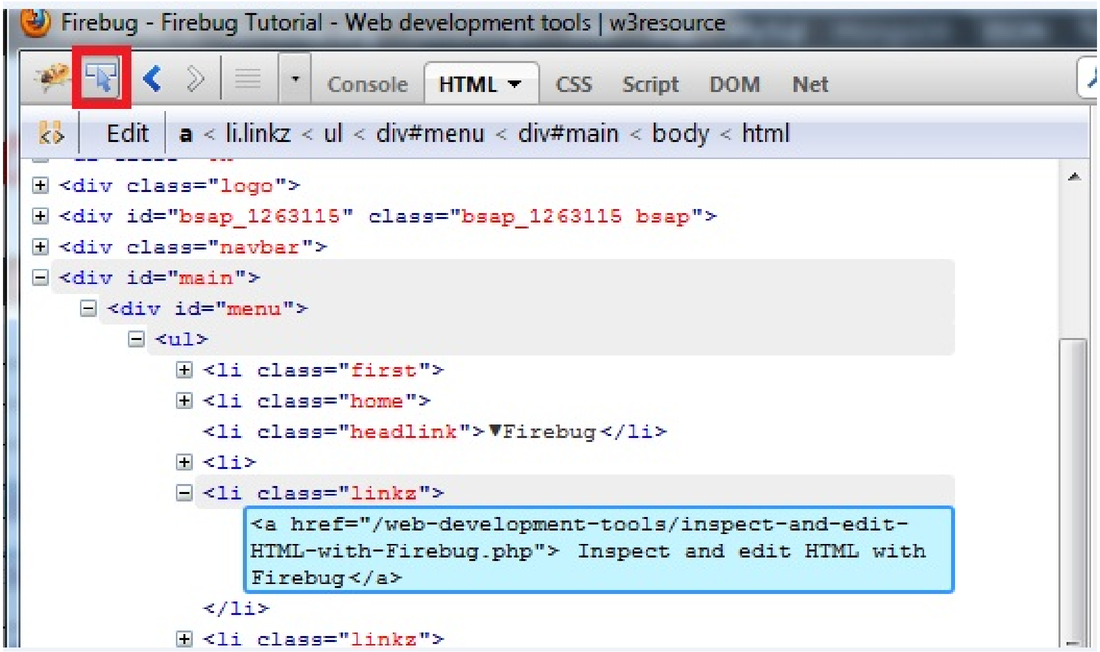


Firebug 组件

Firebug 选项

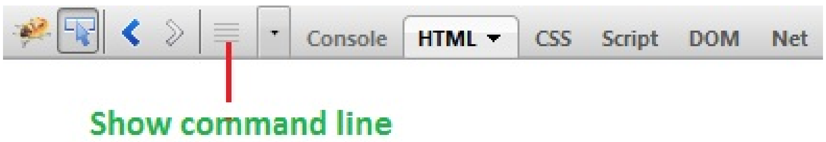


在页面上点击检查元素

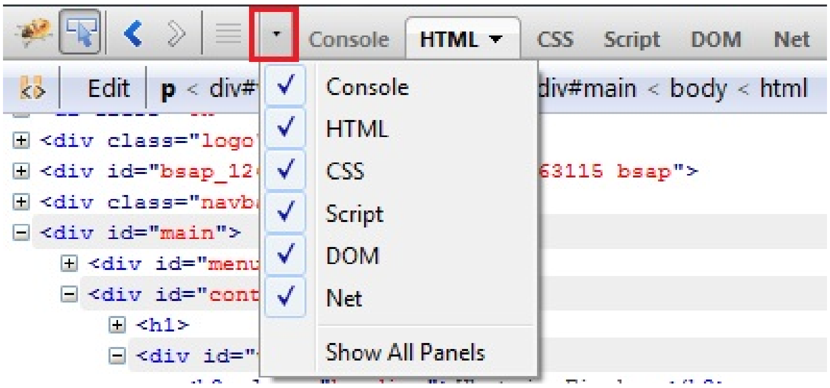


撤销与恢复

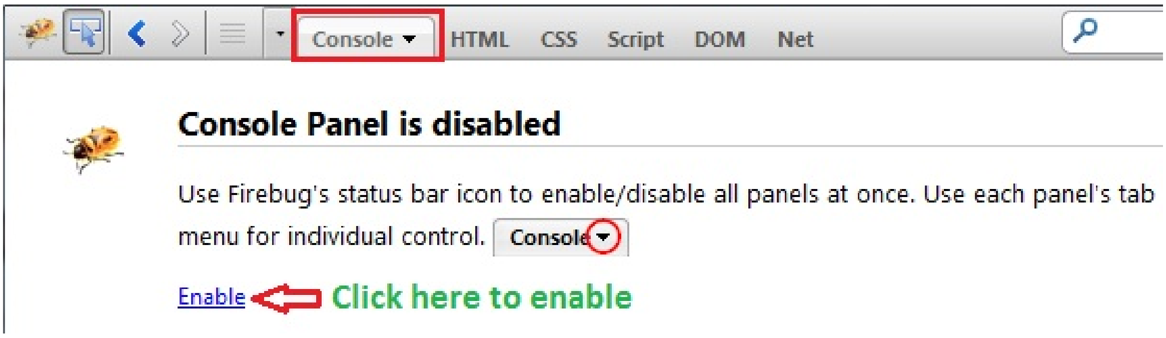
显示命令行



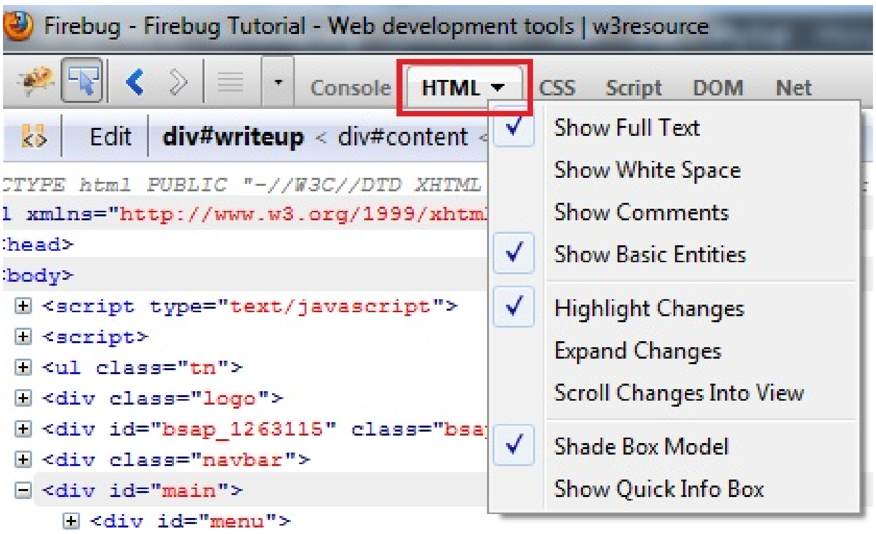
面板选择



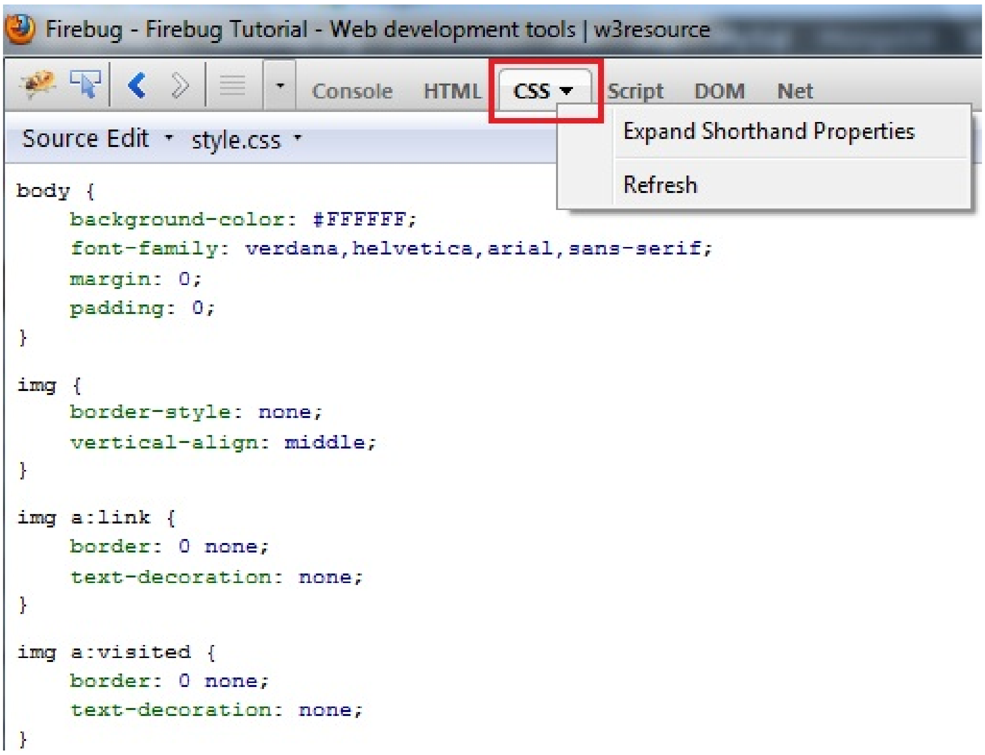
控制台



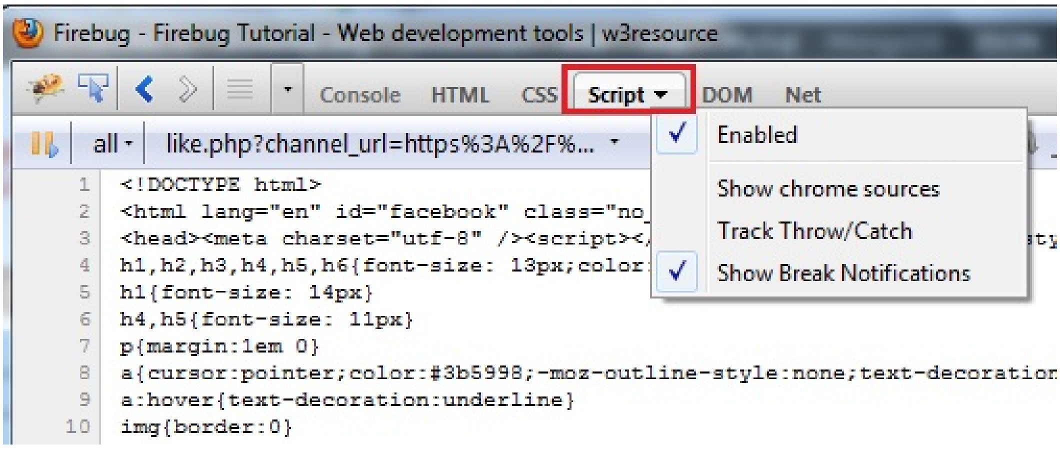
HTML 面板



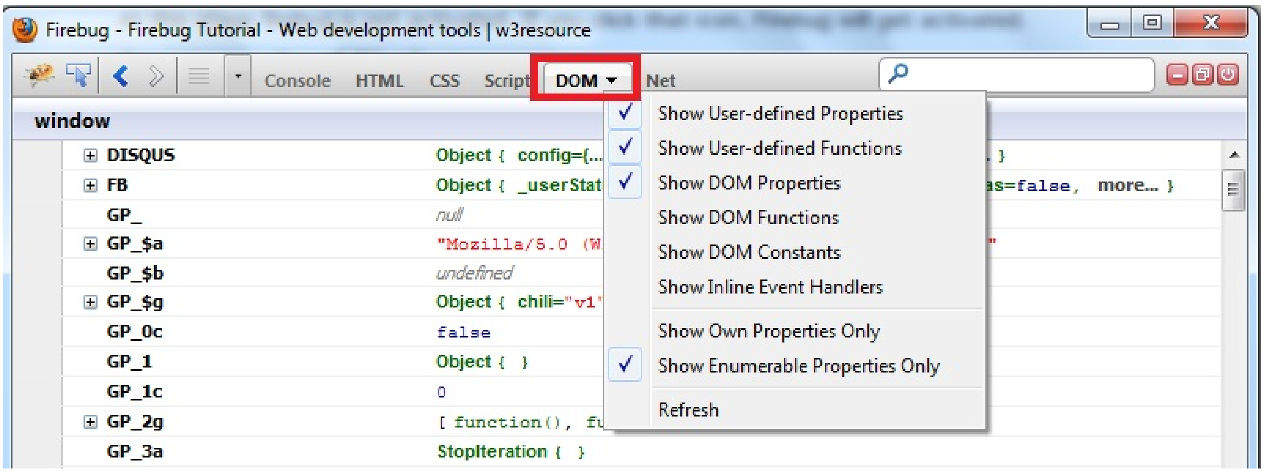
CSS 面板


Firebug - Firebug Tutorial - Web development tools | w3resource
Console
HTML
CSS
Script
DOM
Net
Source Edit
style.css
Expand Shorthand Properties
Refresh
body {
background-color: #FFFFFF;
font-family: verdana,helvetica,arial,sans-serif;
margin: 0;
padding: 0;
}
img {
border-style: none;
vertical-align: middle;
}
img a:link {
border: 0 none;
text-decoration: none;
}
img a:visited {
border: 0 none;
text-decoration: none;
}

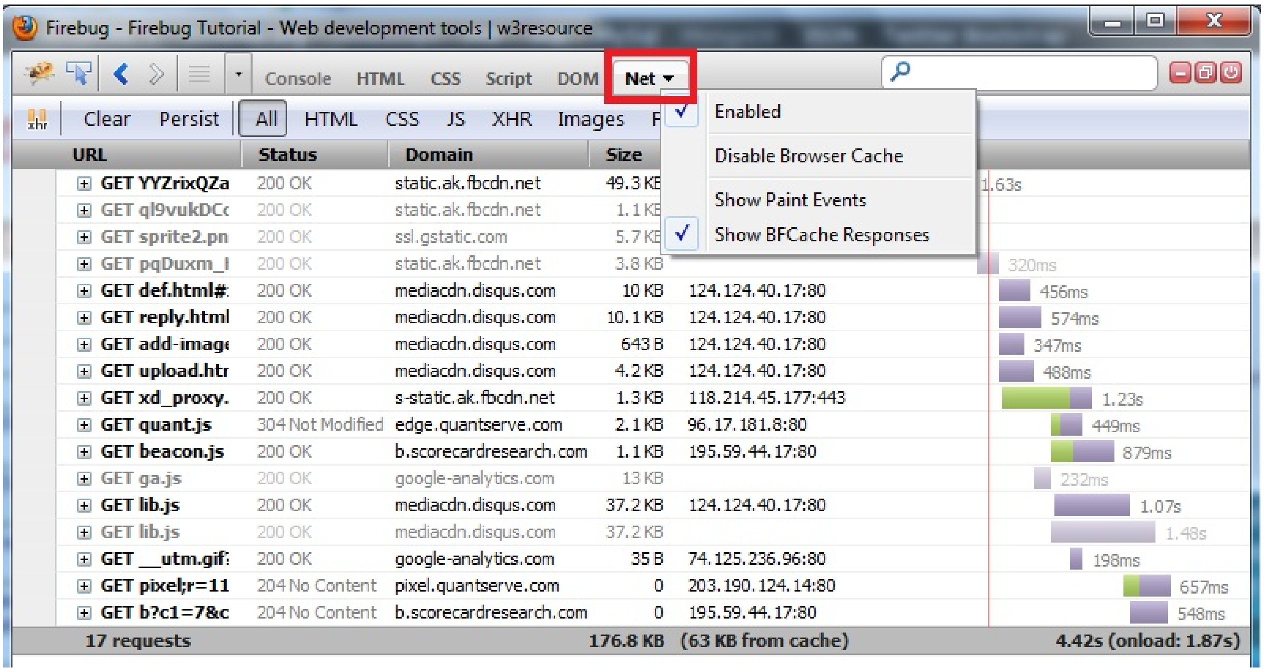
脚本面板


Firebug - Firebug Tutorial - Web development tools | w3resource
Console
HTML
CSS
Script
DOM
Net
all
like.php?channel_url=https%3A%2F%...
Enabled
Show chrome sources
Track Throw/Catch
Show Break Notifications
<!DOCTYPE html>
h1{font-size: 14px}
h4,h5{font-size: 11px}
p{margin:1em 0}
a:hover{text-decoration:underline}
img{border:0}

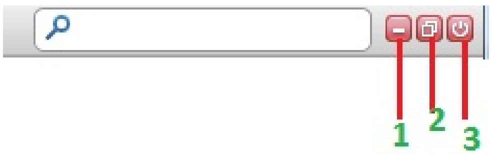
DOM 面板


Firebug - Firebug Tutorial - Web development tools | w3resource
Console
HTML
CSS
Script
DOM
Net
window
Show User-defined Properties
Show User-defined Functions
Show DOM Properties
Show DOM Functions
Show DOM Constants
Show Inline Event Handlers
Show Own Properties Only
Show Enumerable Properties Only
Refresh
DISQUS
FB
GP_
GP_$a
GP_$b
GP_$g
GP_0c
GP_1
GP_1c
GP_2g
GP_3a
null
undefined
false
StopIteration

网络面板 (用于网页测速和优化)



其他按钮



- 1 - 最小化Firebug。
- 2 - 在浏览器窗口安装Firebug。
- 3 - 停用Firebug。

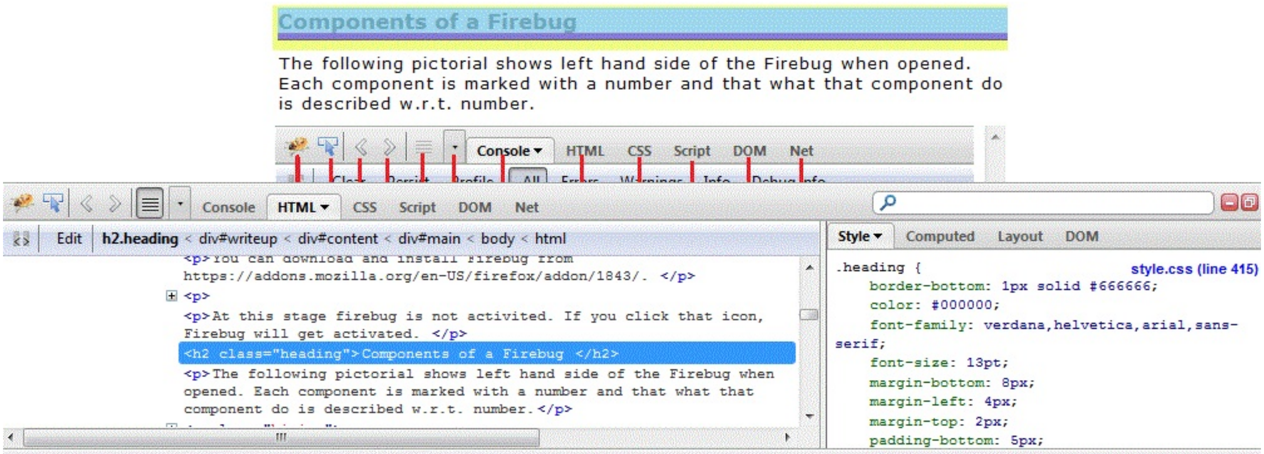
使用Firebug查看和编辑HTML和CSS

描述

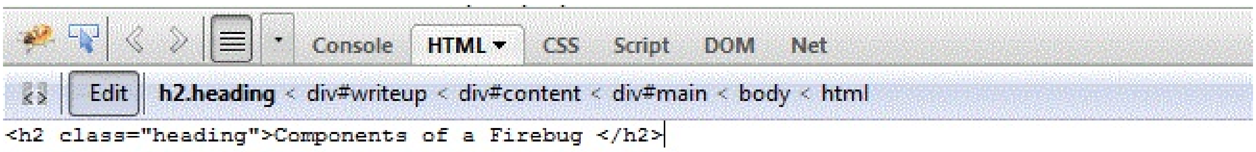
在本章节的教程中，我们将讨论如何使用Firebug查看和编辑HTML和CSS。

使用Firebug查看和编辑HTML

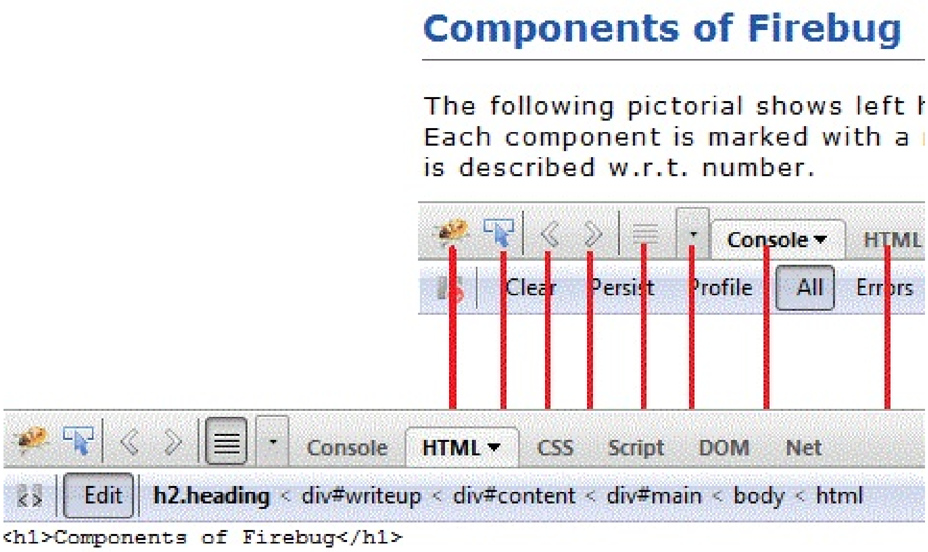
在你要查看的元素上右击鼠标然后点 **Inspect Element(查看元素)**.



现在在Firebug窗口上点击**Edit**（编辑）。



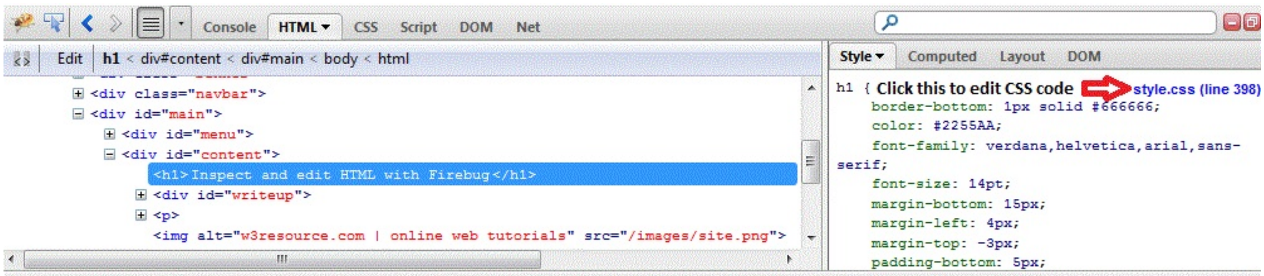
修改代码 <h1>Components of Firebug</h1> .



你只要在Firebug窗口上修改代码就能实时的查看修改后代码在浏览器上的显示效果，你也可以复制代码到你的HTML文件中。

使用Firebug查看和编辑Css

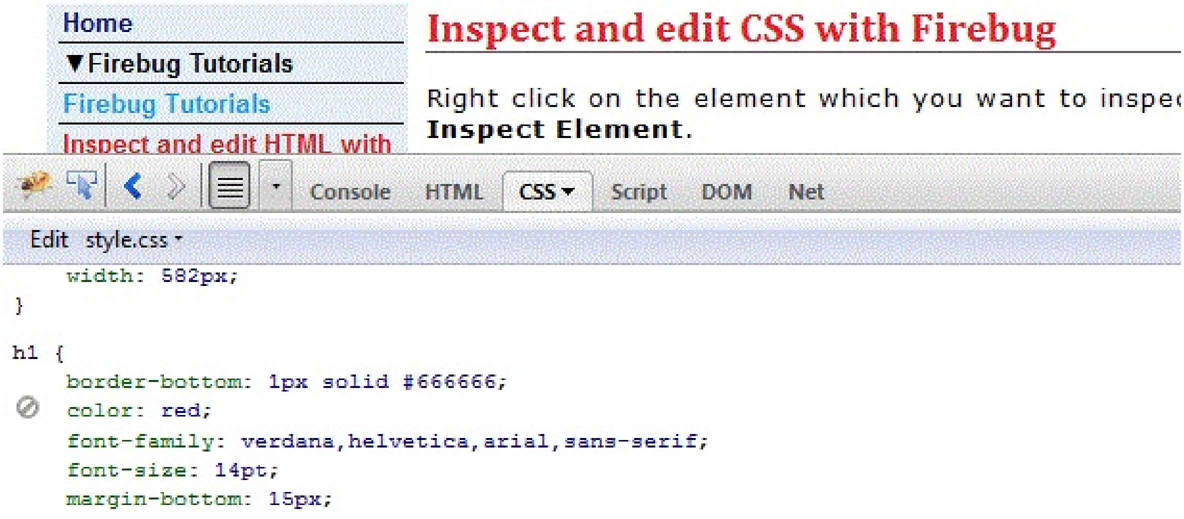
鼠标右击你要查看的元素然后点击**Inspect Element(**查看元素**)**。如下图所示点击样式文件



通过修改代码为 **color:red;** 来修改标题颜色

修改完成后你可以马上看到修改后的效果。

现在你可以复制修改后的样式，取代原有的代码并保存，使之生效：


Home
▼Firebug Tutorials
Firebug Tutorials
Inspect and edit HTML with
Inspect and edit CSS with Firebug
Right click on the element which you want to inspe
Inspect Element.
Console
HTML
CSS
Script
DOM
Net
Edit
style.css
width: 582px;
}
h1 {
border-bottom: 1px solid #666666;
color: red;
font-family: verdana,helvetica,arial,sans-serif;
font-size: 14pt;
margin-bottom: 15px;

使用 Firebug 调试 JavaScript

描述

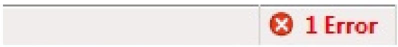
Firebug是一个非常强大的工具，可以帮助您发现代码发现错误的错误并解决错误。

在此我们使用Firebug来处理Javascript代码。

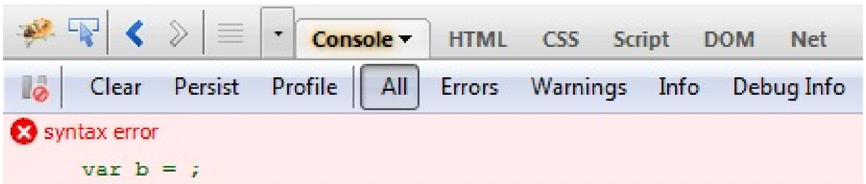
首先我们需要加载页面并打开Firebug。

有时候您需要重新载入页面。

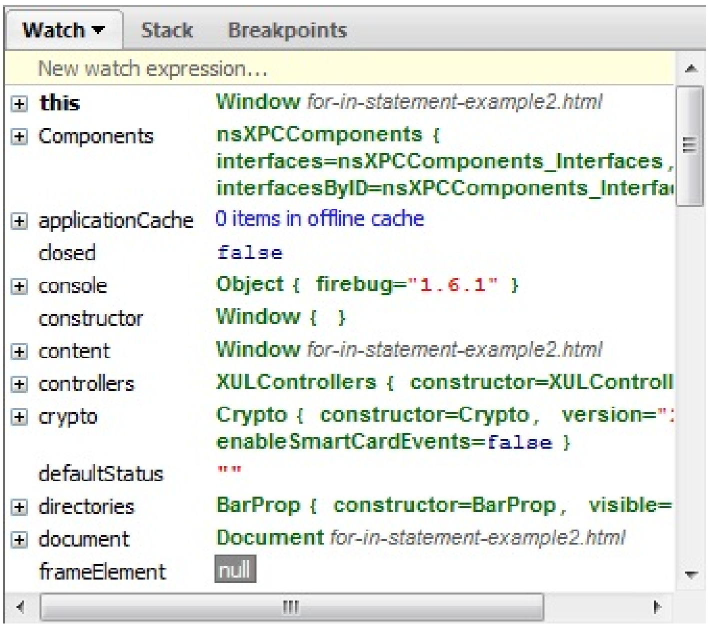
在状态栏的错误数



显示当前页面的错误

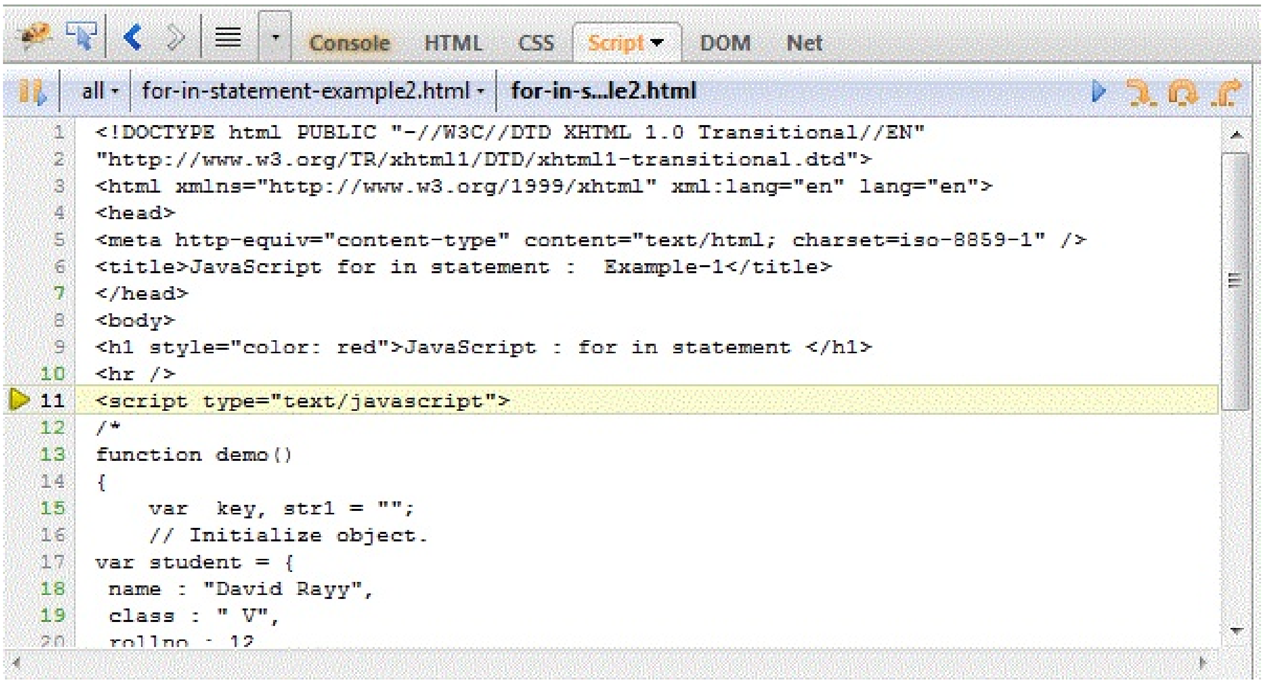


显示错误的详细信息



一步步调试代码

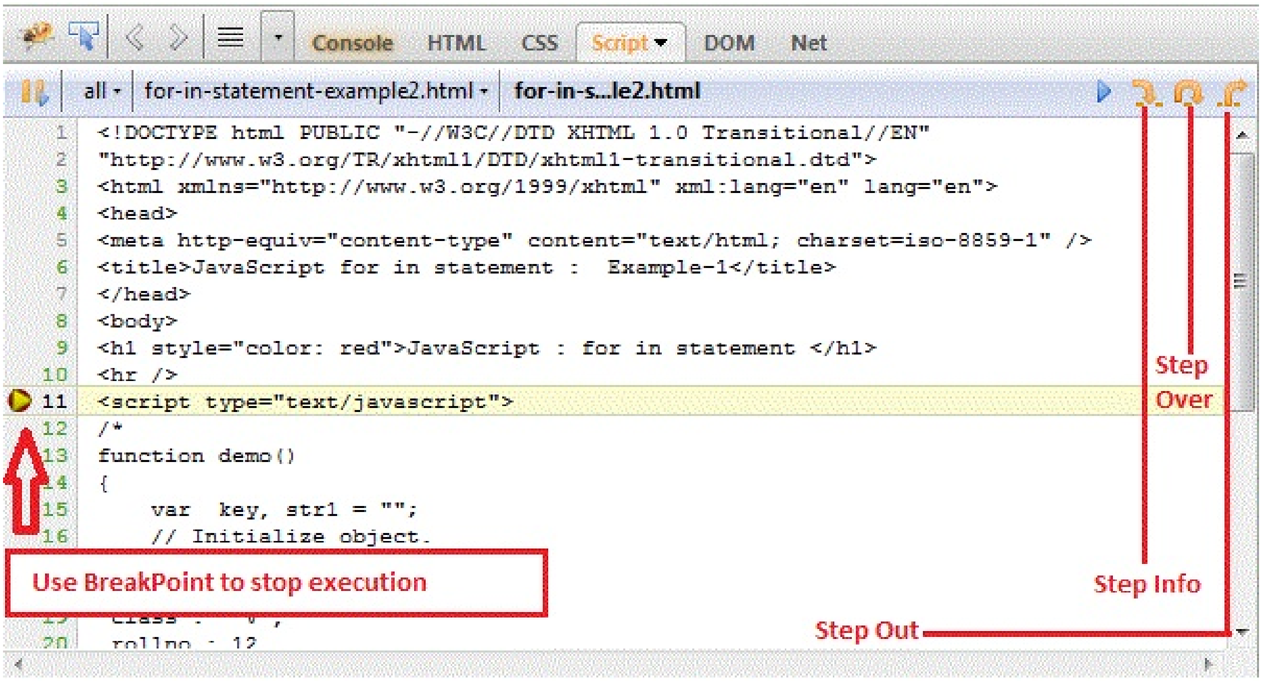
你可以一步步的执行代码。这对代码调试非常有用。



使用断点调试

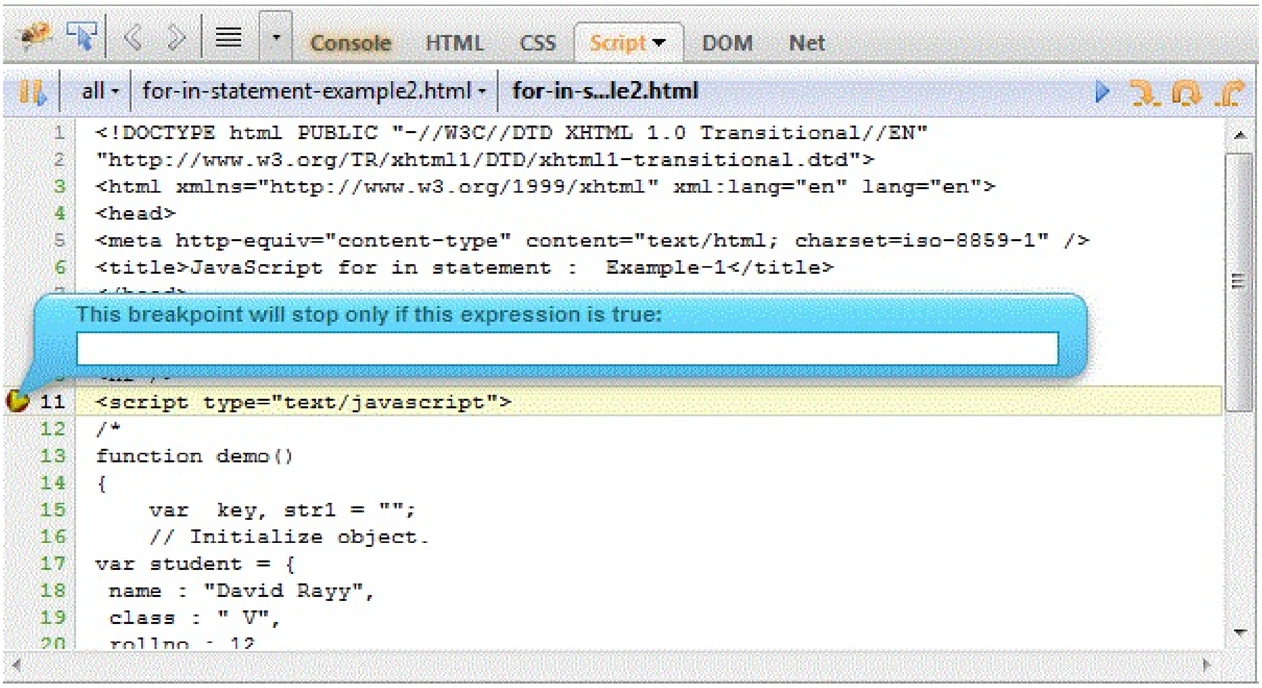
断点调试可以终止代码的执行，你可以通过指定代码范围来查看错误是不是在指定代码范围内。这对于 错误调试很有用。

如果你点击"Step over（单步跳过）" 按钮, Firebug 会更新所有变量直到你在右侧窗口中终止断点执行。



使用表达式让断点工作

你可以写一个表达式，在条件为真时，断点会停止代码的执行。



搜索

你可以使用快速搜索来查找代码中的关键字。



Firebug页面概况查看

使用Firebug的概况，你可以测试Web页面导致延迟加载的文件。

通过打开页面 Firebug > Console（控制台）> Profile（概况）。

你需要重新加载页面，然后点击"概况"查看页面载入情况。



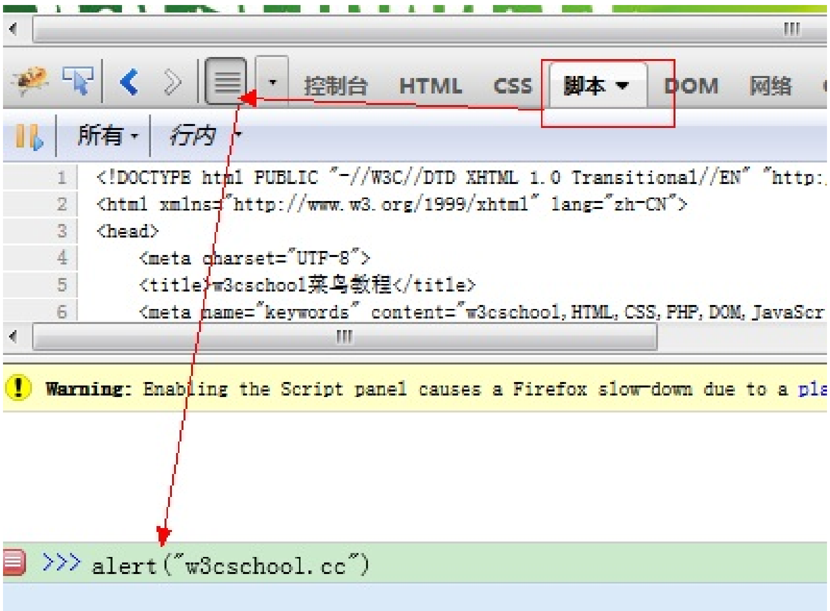
Firebug动态执行JavaScript

您可以使用Firebug来编写并实时执行一个JavaScript。

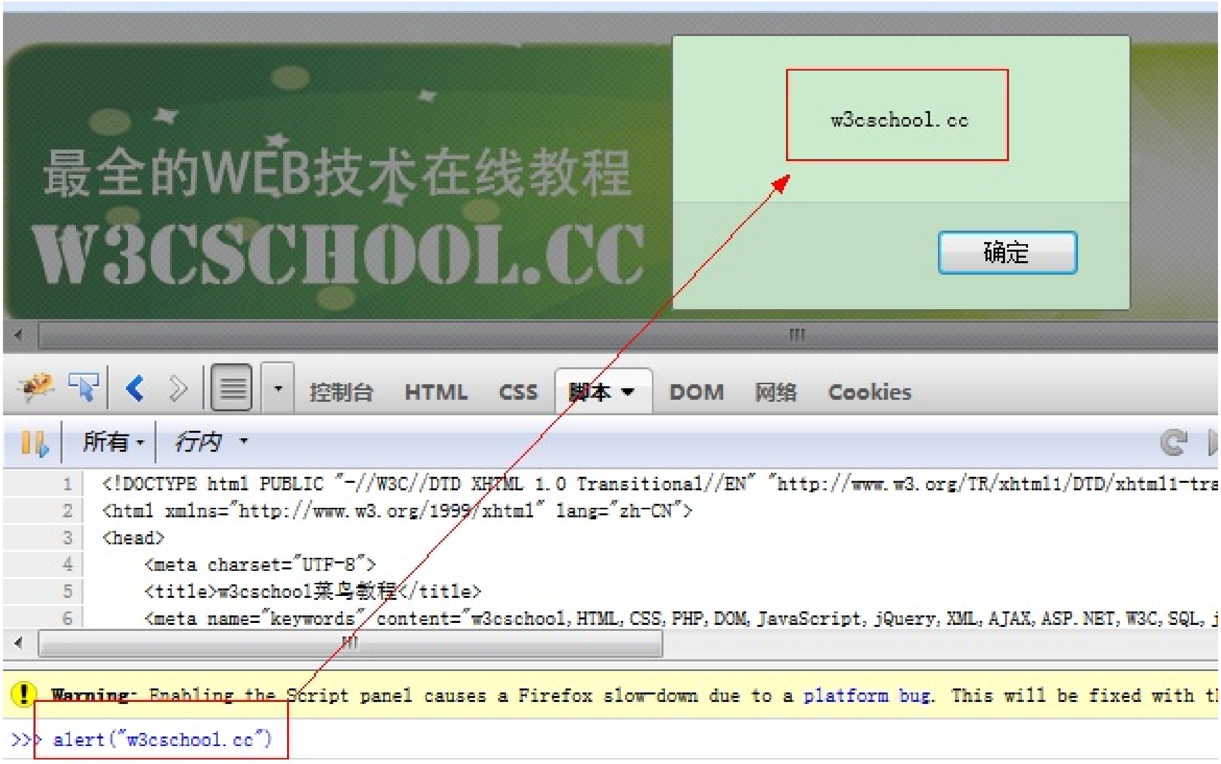
这是为了测试，并确保该脚本工作正常，这是将JavaScript代码部署在生产环境前的好方法。

打开Firebug > Script(脚本) > 显示命令行，代码如下所示：

```
alert("w3cschool.cc")
```



按Enter键后，马上你就能看到代码的输出。如图所示：


最全的WEB技术在线教程
W3CSCHOOL.CC
w3cschool.cc
确定
控制台 HTML CSS 脚本 DOM 网络 Cookies
所有 行内
Warning: Enabling the Script panel causes a Firefox slow-down due to a platform bug. This will be fixed with t
alert("w3cschool.cc")

Firebug记录Javascript日志

你可以使用Firebug来生成日志。

这有助于我们调试web页面并发现页面的错误。

在Firefox浏览器中执行以下代码：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.or
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Firefox测试页面 - w3cschool菜鸟教程</title>
</head>
<body>
<script type="text/javascript">
var a = "w3cschool";
var b = ".cc";

document.write(a,b);
console.log(a + b);
</script>
</body>
</html>
```

使用Firefox浏览器打开以上代码文件firefox-test.html,执行结果及日志记录如下：



Firebug监控网络情况

Firebug可以监控网页中每个文件加载的时间。

打开Firebug。点击"网络"，然后确定已经启用，重新载入当前页面。Firebug显示如下所示：



W3School Bootstrap 教程

来源：Bootstrap 教程

整理：飞龙

Bootstrap 简介

Bootstrap 简介

什么是 Bootstrap？

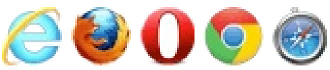
Bootstrap 是一个用于快速开发 Web 应用程序和网站的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的。

历史

Bootstrap 是由 *Twitter* 的 *Mark Otto* 和 *Jacob Thornton* 开发的。Bootstrap 是 2011 年八月在 GitHub 上发布的开源产品。

为什么使用 Bootstrap？

- 移动设备优先：自 Bootstrap 3 起，框架包含了贯穿于整个库的移动设备优先的样式。
- 浏览器支持：所有的主流浏览器都支持 Bootstrap。



- 容易上手：只要您具备 HTML 和 CSS 的基础知识，您就可以开始学习 Bootstrap。
- 响应式设计：Bootstrap 的响应式 CSS 能够自适应于台式机、平板电脑和手机。更多有关响应式设计的内容详见 Bootstrap 响应式设计。



- 它为开发人员创建接口提供了一个简洁统一的解决方案。
- 它包含了功能强大的内置组件，易于定制。
- 它还提供了基于 Web 的定制。
- 它是开源的。

Bootstrap 包的内容

- 基本结构：Bootstrap 提供了一个带有网格系统、链接样式、背景的基本结构。这将在 **Bootstrap** 基本结构 部分详细讲解。
- **CSS**：Bootstrap 自带以下特性：全局的 CSS 设置、定义基本的 HTML 元素样式、可扩展的 class，以及一个先进的网格系统。这将在 **Bootstrap CSS** 部分详细讲解。
- 组件：Bootstrap 包含了十几个可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等。这将在 布局组件 部分详细讲解。
- **JavaScript** 插件：Bootstrap 包含了十几个自定义的 jQuery 插件。您可以直接包含所有的插件，也可以逐个包含这些插件。这将在 **Bootstrap** 插件 部分详细讲解。
- 定制：您可以定制 Bootstrap 的组件、LESS 变量和 jQuery 插件来得到您自己的版本。

在线实例

本站的 Bootstrap 教程包含了上百个实例。

你可以使用我们的在线编辑器在线编辑代码，并点击运行按钮查看结果。

Bootstrap 实例

```
<div class="container">
  <p>Create a responsive table with alternating cell background co

  <div class="table-responsive">
    <table class="table table-striped table-bordered">
      <thead>
        <tr>
          <th>#</th>
          <th>Name</th>
          <th>Street</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Anna Awesome</td>
          <td>Broome Street</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Debbie Dallas</td>
          <td>Houston Street</td>
        </tr>
        <tr>
          <td>3</td>
          <td>John Doe</td>
          <td>Madison Street</td>
        </tr>
      </tbody>
    </table>
  </div>

</div>
```

点击 "尝试一下" 按钮查看它是如何工作的。

Bootstrap 环境安装

Bootstrap 安装是非常容易的。本章将讲解如何下载并安装 Bootstrap，讨论 Bootstrap 文件结构，并通过一个实例演示它的用法。

下载 Bootstrap

您可以从 http://getbootstrap.com/ 上下载 Bootstrap 的最新版本。当您点击这个链接时，您将看到如下所示的网页：



您会看到两个按钮：

- *Download Bootstrap*：下载 Bootstrap。点击该按钮，您可以下载 Bootstrap CSS、JavaScript 和字体的预编译的压缩版本。不包含文档和最初的源代码文件。
- *Download Source*：下载源代码。点击该按钮，您可以直接从 from 上得到最新的 Bootstrap LESS 和 JavaScript 源代码。

如果您使用的是未编译的源代码，您需要编译 LESS 文件来生成可重用的 CSS 文件。对于编译 LESS 文件，Bootstrap 官方只支持 Recess，这是 Twitter 的基于 less.js 的 CSS 提示。

为了更好的了解和更方便的使用，我们将在本教程中使用 Bootstrap 的预编译版本。

由于文件是被编译过和压缩过的，在独立的功能开发中，您不必每次都包含这些独立的文件。

本教程编写时，使用的是最新版（Bootstrap 3）。

文件结构

预编译的 Bootstrap

当您下载了 Bootstrap 的已编译的版本，解压缩 ZIP 文件，您将看到下面的文件/目录结构：

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    └── glyphicons-halflings-regular.woff
```

如上图所示，可以看到已编译的 CSS 和 JS（bootstrap.），以及已编译压缩的 *CSS* 和 *JS*（*bootstrap.min.*）。同时也包含了 Glyphicons 的字体，这是一个可选的 Bootstrap 主题。

Bootstrap 源代码

如果您下载了 Bootstrap 源代码，那么文件结构将如下所示：

```
├── less/
├── js/
├── fonts/
├── dist/
│   ├── css/
│   ├── js/
│   └── fonts/
├── docs-assets/
├── examples/
└── *.html
```

- *less/*、*js/* 和 *fonts/* 下的文件分别是 Bootstrap CSS、JS 和图标字体的源代码。
- *dist/* 文件夹包含了上面预编译下载部分中所列的文件和文件夹。
- *docs-assets/*、*examples/* 和所有的 **.html* 文件是 Bootstrap 文档。

HTML 模板

一个使用了 Bootstrap 的基本的 HTML 模板如下所示：

```
<!DOCTYPE html>
<html>
   <head>
      <title>Bootstrap 模板</title>
      <meta name="viewport" content="width=device-width, initial-sc
      <!-- 引入 Bootstrap -->
      <link href="css/bootstrap.min.css" rel="stylesheet">

      <!-- HTML5 Shim 和 Respond.js 用于让 IE8 支持 HTML5元素和媒体查询
      <!-- 注意： 如果通过 file://  引入 Respond.js 文件，则该文件无法起效
      <!--[if lt IE 9]>
         <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/h
         <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/
      <![endif]-->
   </head>
   <body>
      <h1>Hello, world!</h1>

      <!-- jQuery (Bootstrap 的 JavaScript 插件需要引入 jQuery) -->
      <script src="https://code.jquery.com/jquery.js"></script>
      <!-- 包括所有已编译的插件 -->
      <script src="js/bootstrap.min.js"></script>
   </body>
</html>
```

在这里，您可以看到包含了 **jquery.js**、**bootstrap.min.js** 和 **bootstrap.min.css** 文件，用于让一个常规的 HTML 文件变为使用了 Bootstrap 的模板。

有关上面代码段中每个元素的细节将在 Bootstrap CSS 概览 章节详细讲解。

实例

现在让我们尝试使用使用Bootstrap输出"Hello, world!"：

```
<!DOCTYPE html>
<html>
<head>
   <title>在线尝试 Bootstrap 实例</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

      <h1>Hello, world!</h1>

</body>
</html>
```

Bootstrap CDN推荐

本站实例采用的是百度的静态资源库(http://cdn.code.baidu.com/)上的Bootstrap资源。

百度的静态资源库的 CDN 服务，访问速度更快、加速效果更明显、没有速度和带宽限制、永久免费,引入代码如下：

```
<!-- 新 Bootstrap 核心 CSS 文件 -->
<link href="http://apps.bdimg.com/libs/bootstrap/3.0.3/css/bootstra

<!-- 可选的Bootstrap主题文件（一般不使用） -->
<script src="http://apps.bdimg.com/libs/bootstrap/3.0.3/css/bootstr

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->
<script src="http://apps.bdimg.com/libs/jquery/2.0.0/jquery.min.js'

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
<script src="http://apps.bdimg.com/libs/bootstrap/3.0.3/js/bootstra
```

Bootstrap CSS

Bootstrap CSS 概览

在这一章中，我们将讲解 Bootstrap 底层结构的关键部分，包括我们让 web 开发变得更好、更快、更强壮的最佳实践。

HTML 5 文档类型（Doctype）

Bootstrap 使用了一些 HTML5 元素和 CSS 属性。为了让这些正常工作，您需要使用 HTML5 文档类型（Doctype）。 因此，请在使用 Bootstrap 项目的开头包含下面的代码段。

```
<!DOCTYPE html>
<html>
....
</html>
```

如果在 Bootstrap 创建的网页开头不使用 HTML5 的文档类型（Doctype），您可能会面临一些浏览器显示不一致的问题，甚至可能面临一些特定情境下的不一致，以致于您的代码不能通过 W3C 标准的验证。

移动设备优先

移动设备优先是 Bootstrap 3 的最显著的变化。

在之前的 Bootstrap 版本中（直到 2.x），您需要手动引用另一个 CSS，才能让整个项目友好的支持移动设备。

现在不一样了，Bootstrap 3 默认的 CSS 本身就对移动设备友好支持。

Bootstrap 3 的设计目标是移动设备优先，然后才是桌面设备。这实际上是一个非常及时的转变，因为现在越来越多的用户使用移动设备。

为了让 Bootstrap 开发的网站对移动设备友好，确保适当的绘制和触屏缩放，需要在网页的 head 之中添加 **viewport meta** 标签，如下所示：

```
<meta name="viewport" content="width=device-width, initial-scale=1
```

width 属性控制设备的宽度。假设您的网站将被带有不同屏幕分辨率的设备浏览，那么将它设置为 *device-width* 可以确保它能正确呈现在不同设备上。

initial-scale=1.0 确保网页加载时，以 1:1 的比例呈现，不会有任何的缩放。

在移动设备浏览器上，通过为 **viewport meta** 标签添加 *user-scalable=no* 可以禁用其缩放（zooming）功能。

通常情况下，*maximum-scale=1.0* 与 user-scalable=no 一起使用。这样禁用缩放功能后，用户只能滚动屏幕，就能让您的网站看上去更像原生应用的感觉。

注意，这种方式我们并不推荐所有网站使用，还是要看您自己的情况而定！

```
<meta name="viewport" content="width=device-width,
                                     initial-scale=1.0,
                                     maximum-scale=1.0,
                                     user-scalable=no">
```

响应式图像

```
<img src="..." class="img-responsive" alt="响应式图像">
```

通过添加 *img-responsive* class 可以让 Bootstrap 3 中的图像对响应式布局的支持更友好。

接下来让我们看下这个 class 包含了哪些 css 属性。

在下面的代码中，可以看到*img-responsive* class 为图像赋予了 max-width: 100%; 和 height: auto; 属性，可以让图像按比例缩放，不超过其父元素的尺寸。

```
.img-responsive {
  display: inline-block;
  height: auto;
  max-width: 100%;
}
```

这表明相关的图像呈现为 *inline-block*。当您把元素的 display 属性设置为 inline-block，元素相对于它周围的内容以内联形式呈现，但与内联不同的是，这种情况下我们可以设置宽度和高度。

设置 *height:auto*，相关元素的高度取决于浏览器。

设置 *max-width* 为 100% 会重写任何通过 width 属性指定的宽度。这让图片对响应式布局的支持更友好。

全局显示、排版和链接

基本的全局显示

Bootstrap 3 使用 *body {margin: 0;}* 来移除 body 的边距。

请看下面有关 body 的设置：

```
body {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 14px;
  line-height: 1.428571429;
  color: #333333;
  background-color: #ffffff;
}
```

第一条规则设置 body 的默认字体样式为 *"Helvetica Neue", Helvetica, Arial, sans-serif*。

第二条规则设置文本的默认字体大小为 14 像素。

第三条规则设置默认的行高度为 1.428571429。

第四条规则设置默认的文本颜色为 #333333。

最后一条规则设置默认的背景颜色为白色。

排版

使用 @font-family-base、 @font-size-base 和 @line-height-base 属性作为排版样式。

链接样式

通过属性 @link-color 设置全局链接的颜色。

对于链接的默认样式，如下设置：

```
a:hover,
a:focus {
  color: #2a6496;
  text-decoration: underline;
}

a:focus {
  outline: thin dotted #333;
  outline: 5px auto -webkit-focus-ring-color;
  outline-offset: -2px;
}
```

所以，当鼠标悬停在链接上，或者点击过的链接，颜色会被设置为 #2a6496。同时，会呈现一条下划线。

除此之外，点击过的链接，会呈现一个颜色码为 #333 的细的虚线轮廓。另一条规则是设置轮廓为 5 像素宽，且对于基于 webkit 浏览器有一个 *-webkit-focus-ring-color* 的浏览器扩展。轮廓偏移设置为 -2 像素。

以上所有这些样式都可以在 scaffolding.less 中找到。

避免跨浏览器的不一致

Bootstrap 使用 Normalize 来建立跨浏览器的一致性。

Normalize.css 是一个很小的 CSS 文件，在 HTML 元素的默认样式中提供了更好的跨浏览器一致性。

容器（Container）

```
<div class="container">
  ...
</div>
```

Bootstrap 3 的 *container* class 用于包裹页面上的内容。让我们一起来看看 bootstrap.css 文件中的这个 *.container* class。

```
.container {
   padding-right: 15px;
   padding-left: 15px;
   margin-right: auto;
   margin-left: auto;
}
```

通过上面的代码，把 container 的左右外边距（margin-right、margin-left）交由浏览器决定。

请注意，由于内边距（padding）是固定宽度，默认情况下容器是不可嵌套的。

```
.container:before,
.container:after {
  display: table;
  content: " ";
}
```

这会产生伪元素。设置 *display* 为 *table*，会创建一个匿名的 table-cell 和一个新的块格式化上下文。*:before* 伪元素防止上边距崩塌，*:after* 伪元素清除浮动。

如果 *conteneditable* 属性出现在 HTML 中，由于一些 Opera bug，围绕上述元素创建一个空格。这可以通过使用 *content: " "* 来修复。

```
.container:after {
  clear: both;
}
```

它创建了一个伪元素，并确保了所有的容器包含所有的浮动元素。

Bootstrap 3 CSS 有一个申请响应的媒体查询，在不同的媒体查询阈值范围内都为 container 设置了max-width，用以匹配网格系统。

```
@media (min-width: 768px) {
   .container {
      width: 750px;
}
```

Bootstrap 浏览器/设备支持

Bootstrap 可以在最新的桌面系统和移动端浏览器中很好的工作。

旧的浏览器可能无法很好的支持。

下表为 Bootstrap 支持最新版本的浏览器和平台：

	Chrome	Firefox	IE	Opera	Safari
Android	YES	YES	不适用	NO	不适用
iOS	YES	不适用	不适用	NO	YES
Mac OS X	YES	YES	不适用	YES	YES
Windows	YES	YES	YES*	YES	NO

- Bootstrap 支持 Internet Explorer 8 及更高版本的 IE 浏览器。

Bootstrap 网格系统

本章节我们将讲解 Bootstrap 的网格系统（Grid System）。

Bootstrap 提供了一套响应式、移动设备优先的流式网格系统，随着屏幕或视口（viewport）尺寸的增加，系统会自动分为最多12列。

什么是网格（Grid）？

摘自维基百科：

> “在平面设计中，网格是一种由一系列用于组织内容的相交的直线（垂直的、水平的）组成的结构（通常是二维的）。它广泛应用于打印设计中的设计布局和内容结构。在网页设计中，它是一种用于快速创建一致的布局和有效地使用 HTML 和 CSS 的方法。

简单地说，网页设计中的网格用于组织内容，让网站易于浏览，并降低用户端的负载。

什么是 Bootstrap 网格系统（Grid System）？

Bootstrap 官方文档中有关网格系统的描述：

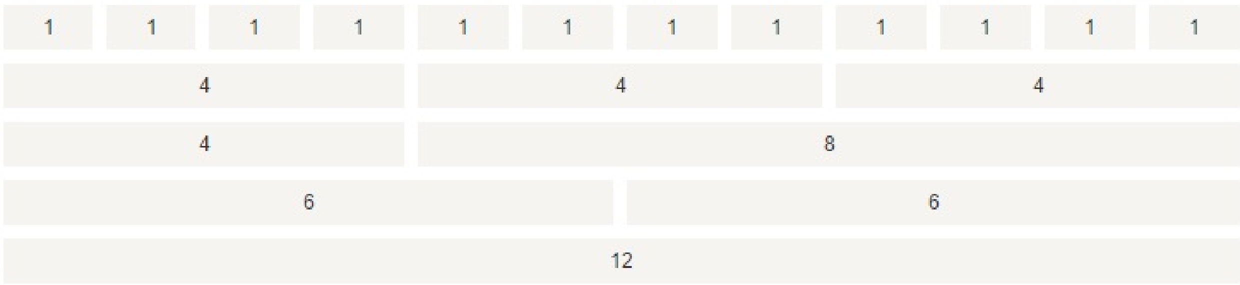
> “Bootstrap 包含了一个响应式的、移动设备优先的、不固定的网格系统，可以随着设备或视口大小的增加而适当地扩展到 12 列。它包含了用于简单的布局选项的预定义类，也包含了用于生成更多语义布局的功能强大的混合类。

让我们来理解一下上面的语句。Bootstrap 3 是移动设备优先的，在这个意义上，Bootstrap 代码从小屏幕设备（比如移动设备、平板电脑）开始，然后扩展到大屏幕设备（比如笔记本电脑、台式电脑）上的组件和网格。

移动设备优先策略

- 内容
  - 决定什么是最重要的。
- 布局
  - 优先设计更小的宽度。
  - 基础的 CSS 是移动设备优先，媒体查询是针对于平板电脑、台式电脑。
- 渐进增强
  - 随着屏幕大小的增加而添加元素。

相应式的网格系统随着屏幕或视口（viewport）尺寸的增加，系统会自动分为最多12列。



Bootstrap 网格系统（Grid System）的工作原理

网格系统通过一系列包含内容的行和列来创建页面布局。下面列出了 Bootstrap 网格系统是如何工作的：

- 行必须放置在 **.container** class 内，以便获得适当的对齐（alignment）和内边距（padding）。
- 使用行来创建列的水平组。
- 内容应该放置在列内，且唯有列可以是行的直接子元素。
- 预定义的网格类，比如 **.row** 和 **.col-xs-4**，可用于快速创建网格布局。LESS 混合类可用于更多语义布局。
- 列通过内边距（padding）来创建列内容之间的间隙。该内边距是通过 **.rows** 上的外边距（margin）取负，表示第一列和最后一列的行偏移。
- 网格系统是通过指定您想要横跨的十二个可用的列来创建的。例如，要创建三个相等的列，则使用三个 **.col-xs-4**。

媒体查询

媒体查询是非常别致的"有条件的 CSS 规则"。它只适用于一些基于某些规定条件的 CSS。如果满足那些条件，则应用相应的样式。

Bootstrap 中的媒体查询允许您基于视口大小移动、显示并隐藏内容。下面的媒体查询在 LESS 文件中使用，用来创建 Bootstrap 网格系统中的关键的分界点阈值。

```
/* 超小设备（手机，小于 768px） */
/* Bootstrap 中默认情况下没有媒体查询 */

/* 小型设备（平板电脑，768px 起） */
@media (min-width: @screen-sm-min) { ... }

/* 中型设备（台式电脑，992px 起） */
@media (min-width: @screen-md-min) { ... }

/* 大型设备（大台式电脑，1200px 起） */
@media (min-width: @screen-lg-min) { ... }
```

我们有时候也会在媒体查询代码中包含 **max-width**，从而将 CSS 的影响限制在更小范围的屏幕大小之内。

```
@media (max-width: @screen-xs-max) { ... }
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max)
@media (min-width: @screen-md-min) and (max-width: @screen-md-max)
@media (min-width: @screen-lg-min) { ... }
```

媒体查询有两个部分，先是一个设备规范，然后是一个大小规则。在上面的案例中，设置了下列的规则：

让我们来看下面这行代码：

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max)
```

对于所有带有 *min-width: @screen-sm-min* 的设备，如果屏幕的宽度小于 *@screen-sm-max*，则会进行一些处理。

网格选项

下表总结了 Bootstrap 网格系统如何跨多个设备工作：

	超小设备收集（<768px）	小型设备平板电脑（≥768px）	中型设备台式电脑（≥992px）	大型设备台式电脑（≥1200px）
网格行为	一直是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的
最大容器宽度	None (auto)	750px	970px	1170px
Class 前缀	**.col-xs-**	**.col-sm-**	**.col-md-**	**.col-lg-**
列 #	12	12	12	12
最大列宽	Auto	60px	78px	95px
间隙宽度	30px（一个列的每边分别 15px）	30px（一个列的每边分别 15px）	30px（一个列的每边分别 15px）	30px（一个列的每边分别 15px）
可嵌套	Yes	Yes	Yes	Yes
偏移量	Yes	Yes	Yes	Yes
列排序	Yes	Yes	Yes	Yes

基本的网格结构

下面是 Bootstrap 网格的基本结构：

```
<div class="container">
   <div class="row">
      <div class="col-*-*"></div>
      <div class="col-*-*"></div>
   </div>
   <div class="row">...</div>
</div>
<div class="container">....
```

让我们来看几个简单的网格实例：

- 实例：堆叠的水平
- 实例：中型和大型设备

- 实例：手机、平板电脑、台式电脑

响应式的列重置

以下实例包含了4个网格，但是我们在小设备浏览时无法确定网格显示的位置。

为了解决这个问题，使用 可以使用 **.clearfix** class和 响应式实用工具来解决，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 响应式的列重置</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
   <div class="row" >
      <div class="col-xs-6 col-sm-3"
         style="background-color: #dedef8;
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         <p>Lorem ipsum dolor sit amet, consectetur adipisicing eli
      </div>
      <div class="col-xs-6 col-sm-3"
         style="background-color: #dedef8;box-shadow:
         inset 1px -1px 1px #444, inset -1px 1px 1px #444;">
         <p>Lorem ipsum dolor sit amet, consectetur adipisicing eli
            eiusmod tempor incididunt ut labore et dolore magna ali
            enim ad minim veniam, quis nostrud exercitation ullamco
            nisi ut aliquip ex ea commodo consequat.
         </p>
         <p>Lorem ipsum dolor sit amet, consectetur adipisicing eli
            eiusmod tempor incididunt ut.
         </p>
      </div>

      <div class="clearfix visible-xs"></div>

      <div class="col-xs-6 col-sm-3"
         style="background-color: #dedef8;
         box-shadow:inset 1px -1px 1px #444, inset -1px 1px 1px #44
         <p>Ut enim ad minim veniam, quis nostrud exercitation ulla
            laboris nisi ut aliquip ex ea commodo consequat.
         </p>
      </div>
      <div class="col-xs-6 col-sm-3"
         style="background-color: #dedef8;box-shadow:
         inset 1px -1px 1px #444, inset -1px 1px 1px #444;">
```

```
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing eli
               eiusmod tempor incididunt ut labore et dolore magna ali
               enim ad minim
            </p>
         </div>
      </div>
   </div>

   </body>
   </html>
```

浏览器上调整窗口大小查看变化，或在您手机上查看效果。

偏移列

偏移是一个用于更专业的布局的有用功能。它们可用来给列腾出更多的空间。例如，**.col-xs=*** 类不支持偏移，但是它们可以简单地通过使用一个空的单元格来实现该效果。

为了在大屏幕显示器上使用偏移，请使用 **.col-md-offset-*** 类。这些类会把一个列的左外边距（margin）增加 * 列，其中 * 范围是从 **1** 到 **11**。

在下面的实例中，我们有 <div class="col-md-6">..</div>，我们将使用 **.col-md-offset-3** class 来居中这个 div。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 偏移列</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">

   <h1>Hello, world!</h1>

   <div class="row" >
      <div class="col-xs-6 col-md-offset-3"
         style="background-color: #dedef8;box-shadow:
         inset 1px -1px 1px #444, inset -1px 1px 1px #444;">
         <p>Lorem ipsum dolor sit amet, consectetur adipisicing
            elit.
         </p>
      </div>

   </div>
</div>

</body>
</html>
```

结果如下所示：



嵌套列

为了在内容中嵌套默认的网格，请添加一个新的 **.row**，并在一个已有的 **.col-md-*** 列内添加一组 **.col-md-*** 列。被嵌套的行应包含一组列，这组列个数不能超过 12（其实，没有要求你必须占满12列）。

在下面的实例中，布局有两个列，第二列被分为两行四个盒子。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 嵌套列</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
```

```
    <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">

    <h1>Hello, world!</h1>

    <div class="row">

        <div class="col-md-3" style="background-color: #dedef8;box-sh
            inset 1px -1px 1px #444, inset -1px 1px 1px #444;">
            <h4>第一列</h4>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing eli
        </div>

        <div class="col-md-9" style="background-color: #dedef8;box-sh
            inset 1px -1px 1px #444, inset -1px 1px 1px #444;">
            <h4>第二列 - 分为四个盒子</h4>
            <div class="row">
                <div class="col-md-6" style="background-color: #B18904;
                    box-shadow: inset 1px -1px 1px #444, inset -1px 1px
                    <p>Consectetur art party Tonx culpa semiotics. Pinte
                        assumenda minim organic quis.
                    </p>
                </div>
                <div class="col-md-6" style="background-color: #B18904;
                    box-shadow: inset 1px -1px 1px #444, inset -1px 1px
                    <p> sed do eiusmod tempor incididunt ut labore et do
                        aliqua. Ut enim ad minim veniam, quis nostrud exe
                        ullamco laboris nisi ut aliquip ex ea commodo con
                    </p>
                </div>
            </div>

            <div class="row">
                <div class="col-md-6" style="background-color: #B18904;
                    box-shadow: inset 1px -1px 1px #444, inset -1px 1px
                    <p>quis nostrud exercitation ullamco laboris nisi ut
                        aliquip ex ea commodo consequat.
                    </p>
                </div>
                <div class="col-md-6" style="background-color: #B18904;
                    box-shadow: inset 1px -1px 1px #444, inset -1px 1px
                    <p>Lorem ipsum dolor sit amet, consectetur adipisici
                        sed do eiusmod tempor incididunt ut labore et dol
                        aliqua. Ut enim ad minim.</p>
                </div>
            </div>

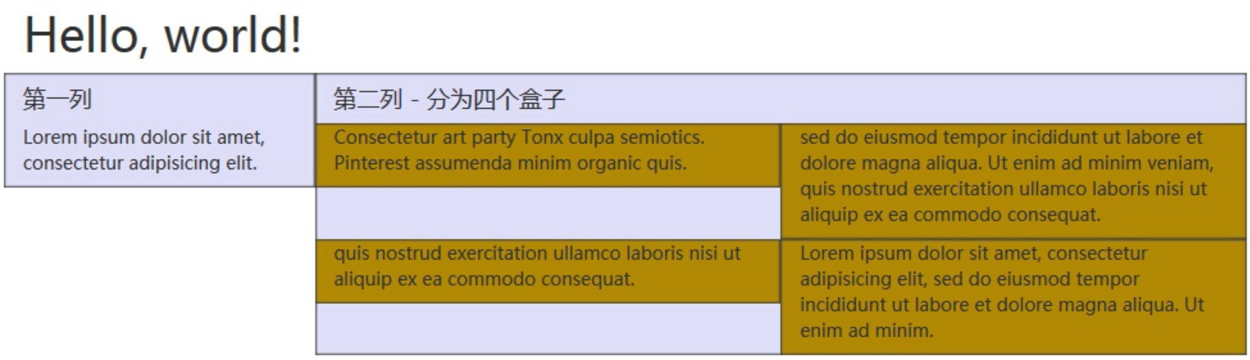
        </div>

    </div>
```

```
</div>

</body>
</html>
```

结果如下所示：



列排序

Bootstrap 网格系统另一个完美的特性，就是您可以很容易地以一种顺序编写列，然后以另一种顺序显示列。

您可以很轻易地改变带有 **.col-md-push-*** 和 **.col-md-pull-*** 类的内置网格列的顺序，其中 * 范围是从 **1** 到 **11**。

在下面的实例中，我们有两列布局，左列很窄，作为侧边栏。我们将使用 **.col-md-push-*** 和 **.col-md-pull-*** 类来互换这两列的顺序。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 列排序</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">

   <h1>Hello, world!</h1>

   <div class="row">
      <p>排序前</p>
      <div class="col-md-4" style="background-color: #dedef8;
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         我在左边
      </div>
      <div class="col-md-8" style="background-color: #dedef8;
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         我在右边
      </div>
   </div><br>
   <div class="row">
      <p>排序后</p>
      <div class="col-md-4 col-md-push-8"
         style="background-color: #dedef8;
         box-shadow: inset 1px -1px 1px #444,
         inset -1px 1px 1px #444;">
         我在左边
      </div>
      <div class="col-md-8 col-md-pull-4"
         style="background-color: #dedef8;
         box-shadow: inset 1px -1px 1px #444,
         inset -1px 1px 1px #444;">
         我在右边
      </div>
   </div>

</div>

</body>
</html>
```

结果如下所示：

Hello, world!

排序前

我在左边	我在右边

排序后

我在右边	我在左边

Bootstrap 排版

Bootstrap 使用 Helvetica Neue、 Helvetica、 Arial 和 sans-serif 作为其默认的字体栈。

使用 Bootstrap 的排版特性，您可以创建标题、段落、列表及其他内联元素。

标题

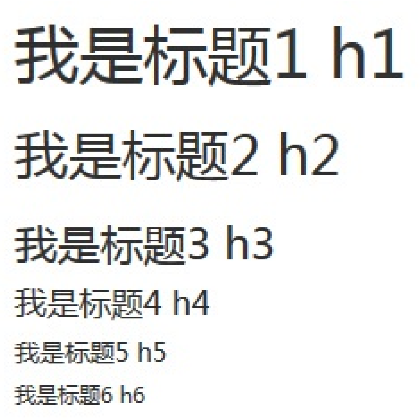
Bootstrap 中定义了所有的 HTML 标题（h1 到 h6）的样式。请看下面的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标题</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h1>我是标题1 h1</h1>
<h2>我是标题2 h2</h2>
<h3>我是标题3 h3</h3>
<h4>我是标题4 h4</h4>
<h5>我是标题5 h5</h5>
<h6>我是标题6 h6</h6>

</body>
</html>
```

结果如下所示：



内联子标题

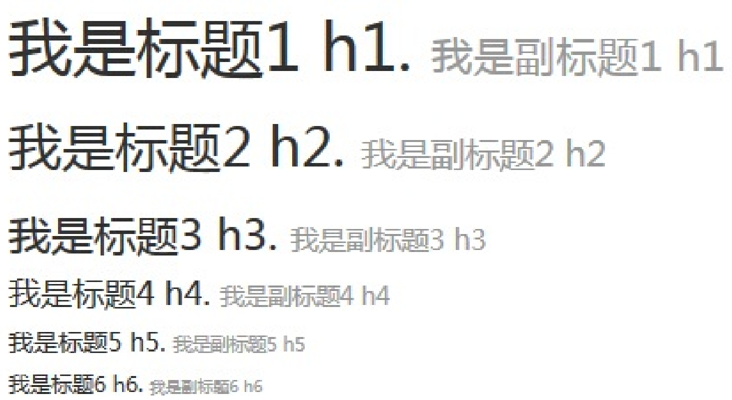
如果需要向任何标题添加一个内联子标题，只需要简单地在元素两旁添加 <small>，或者添加 **.small** class，这样子您就能得到一个字号更小的颜色更浅的文本，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 内联子标题</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h1>我是标题1 h1\. <small>我是副标题1 h1</small></h1>
<h2>我是标题2 h2\. <small>我是副标题2 h2</small></h2>
<h3>我是标题3 h3\. <small>我是副标题3 h3</small></h3>
<h4>我是标题4 h4\. <small>我是副标题4 h4</small></h4>
<h5>我是标题5 h5\. <small>我是副标题5 h5</small></h5>
<h6>我是标题6 h6\. <small>我是副标题6 h6</small></h6>

</body>
</html>
```

结果如下所示：



引导主体副本

为了给段落添加强调文本，则可以添加 class="lead"，这将得到更大更粗、行高更高的文本，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 引导主体副本</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
<hr>
<h2>引导主体副本</h2>
<p class="lead">这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本

</body>
</html>
```

结果如下所示：

引导主体副本

这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。这是一个演示引导主体副本用法的实例。

强调

HTML 的默认强调标签 <small>（设置文本为父文本大小的 85%）、<strong>（设置文本为更粗的文本）、<em>（设置文本为斜体）。

Bootstrap 提供了一些用于强调文本的类，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 强调</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<small>本行内容是在标签内</small><br>
<strong>本行内容是在标签内</strong><br>
<em>本行内容是在标签内，并呈现为斜体</em><br>
<p class="text-left">向左对齐文本</p>
<p class="text-center">居中对齐文本</p>
<p class="text-right">向右对齐文本</p>
<p class="text-muted">本行内容是减弱的</p>
<p class="text-primary">本行内容带有一个 warning class</p>
<p class="text-success">本行内容带有一个 success class</p>
<p class="text-info">本行内容带有一个 info class</p>
<p class="text-warning">本行内容带有一个 warning class</p>
<p class="text-danger">本行内容带有一个 danger class</p>

</body>
</html>
```

结果如下所示：

缩写

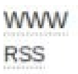
HTML <abbr>元素提供了用于缩写的标记，比如 WWW 或 HTTP。Bootstrap 定义 <abbr> 元素的样式为显示在文本底部的一条虚线边框，当鼠标悬停在上面时会显示完整的文本（只要您为 <abbr> title 属性添加了文本）。为了得到一个更小字体的文本，请添加 .initialism 到 <abbr>。</abbr>

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 缩写</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<abbr title="World Wide Web">WWW</abbr><br>
<abbr title="Real Simple Syndication" class="initialism">RSS</abbr>

</body>
</html>
```

结果如下所示：



地址（Address）

使用 <address> 标签，您可以在网页上显示联系信息。由于 <address> 默认为 display:block;，您需要使用 标签来为封闭的地址文本添加换行。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 地址</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<address>
  <strong>Some Company, Inc.</strong><br>
  007 street<br>
  Some City, State XXXXX<br>
  <abbr title="Phone">P:</abbr> (123) 456-7890
</address>

<address>
  <strong>Full Name</strong><br>
  <a href="mailto:#">mailto@somedomain.com</a>
</address>

</body>
</html>
```

结果如下所示：

Some Company, Inc.
007 street
Some City, State XXXXX
P: (123) 456-7890

Full Name
mailto@somedomain.com

引用（Blockquote）

您可以在任意的 HTML 文本旁使用默认的 <blockquote>。其他选项包括，添加一个 <small> 标签来标识引用的来源，使用 class *.pull-right* 向右对齐引用。下面的实例演示了这些特性：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 引用</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<blockquote><p>
这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。这是一个
<blockquote>这是一个带有源标题的引用。<small>Someone famous in <cite ti
<blockquote class="pull-right">这是一个向右对齐的引用。<small>Someone f

</body>
</html>
```

结果如下所示：

> 这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。这是一个默认的引用实例。

> 这是一个带有源标题的引用。
> — Someone famous in Source Title

> 这是一个向右对齐的引用。
> Someone famous in Source Title —

列表

Bootstrap 支持有序列表、无序列表和定义列表。

- 有序列表：有序列表是指以数字或其他有序字符开头的列表。
- 无序列表：无序列表是指没有特定顺序的列表，是以传统风格的着重号开头的列表。如果您不想显示这些着重号，您可以使用 class *.list-unstyled* 来移除样式。您也可以通过使用 class *.list-inline* 把所有的列表项放在同一行中。
- 定义列表：在这种类型的列表中，每个列表项可以包含 <dt> 和 <dd> 元素。<dt> 代表 定义术语，就像字典，这是被定义的属于（或短语）。接着，<dd> 是 <dt> 的描述。您可以使用 class *dl-horizontal* 把 <dl> 行中的属于与描述并排显示。

下面的实例演示了这些类型的列表：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 列表</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h4>有序列表</h4>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ol>
<h4>无序列表</h4>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>
<h4>未定义样式列表</h4>
<ul class="list-unstyled">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>
<h4>内联列表</h4>
<ul class="list-inline">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>
<h4>定义列表</h4>
<dl>
  <dt>Description 1</dt>
  <dd>Item 1</dd>
  <dt>Description 2</dt>
  <dd>Item 2</dd>
</dl>
<h4>水平的定义列表</h4>
<dl class="dl-horizontal">
  <dt>Description 1</dt>
  <dd>Item 1</dd>
  <dt>Description 2</dt>
  <dd>Item 2</dd>
</dl>
```

```
</body>
</html>
```

结果如下所示：

有序列表

1. Item 1
2. Item 2
3. Item 3
4. Item 4

无序列表

- Item 1
- Item 2
- Item 3
- Item 4

未定义样式列表

Item 1
Item 2
Item 3
Item 4

内联列表

Item 1 Item 2 Item 3 Item 4

定义列表

Description 1
Item 1
Description 2
Item 2

水平的定义列表

Description 1 Item 1
Description 2 Item 2

更多排版类

下表提供了 Bootstrap 更多排版类的实例：

类	描述	实例
.lead	使段落突出显示	尝试一下
		尝试

		一下
.text-left	设定文本左对齐	尝试一下
.text-center	设定文本居中对齐	尝试一下
.text-right	设定文本右对齐	尝试一下
.text-justify	设定文本对齐,段落中超出屏幕部分文字自动换行	尝试一下
.text-nowrap	段落中超出屏幕部分不换行	尝试一下
.text-lowercase	设定文本小写	尝试一下
.text-uppercase	设定文本大写	尝试一下
.text-capitalize	设定单词首字母大写	尝试一下
.initialism	显示在 <abbr> 元素中的文本以小号字体展示	尝试一下
.blockquote-reverse	设定引用右对齐	尝试一下

		下
.list-unstyled	移除默认的列表样式，列表项中左对齐 (和 中)。 这个类仅适用于直接子列表项 (如果需要移除嵌套的列表项，你需要在嵌套的列表中使用该样式)	尝试一下
.list-inline	将所有列表项放置同一行	尝试一下
.dl-horizontal	该类设置了浮动和偏移，应用于 <dl> 元素和 <dt> 元素中，具体实现可以查看实例	尝试一下
.pre-scrollable	使 <pre> 元素可滚动 scrollable	尝试一下

Bootstrap 代码

Bootstrap 允许您以两种方式显示代码：

- 第一种是 <code> 标签。如果您想要内联显示代码，那么您应该使用 <code> 标签。
- 第二种是 <pre> 标签。如果代码需要被显示为一个独立的块元素或者代码有多行，那么您应该使用 <pre> 标签。

请确保当您使用 <pre> 和 <code> 标签时，开始和结束标签使用了 unicode 变体：**<** 和 **>**。

让我们来看看下面的实例：

实例

```
<!DOCTYPE html>
<html>
<head>
  <title>Bootstrap 实例 - 代码</title>
  <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <script src="/scripts/jquery.min.js"></script>
  <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p><code>&lt;header&gt;</code> 作为内联元素被包围。</p>
<p>如果需要把代码显示为一个独立的块元素，请使用 <pre> 标签：</p>
<pre>
  &lt;article&gt;
    &lt;h1&gt;Article Heading&lt;/h1&gt;
  &lt;/article&gt;
</pre>
</body>
</html>
```

实例展示如下图：



Bootstrap 表格

Bootstrap 提供了一个清晰的创建表格的布局。下表列出了 Bootstrap 支持的一些表格元素：

标签	描述
<table>	容纳以表格形式显示数据的元素。
<thead>	表格标题行的容器元素（<tr>），用来标识表格列。
<tbody>	表格主体中的表格行的容器元素（<tr>）。
<tr>	一组出现在单行上的表格单元格的容器元素（<td> 或 <th>）。
<td>	默认的表格单元格。
<th>	特殊的表格单元格，用来标识列或行（取决于范围和位置）。必须在 <thead> 内使用。
<caption>	关于表格存储内容的描述或总结。

基本的表格

如果您想要一个只带有内边距（padding）和水平分割的基本表，请添加 class *.table*，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本的表格</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<table class="table">
   <caption>基本的表格布局</caption>
   <thead>
      <tr>
         <th>名称</th>
         <th>城市</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Tanmay</td>
         <td>Bangalore</td>
      </tr>
      <tr>
         <td>Sachin</td>
         <td>Mumbai</td>
      </tr>
   </tbody>
</table>

</body>
</html>
```

结果如下所示：

基本的表格布局

名称	城市
Tanmay	Bangalore
Sachin	Mumbai

可选的表格类

除了基本的表格标记和 .table class，还有一些可以用来为标记定义样式的类。下面将向您介绍这些类。

条纹表格

通过添加 *.table-striped* class，您将在 <tbody> 内的行上看到条纹，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 条纹表格</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<table class="table table-striped">
   <caption>条纹表格布局</caption>
   <thead>
      <tr>
         <th>名称</th>
         <th>城市</th>
         <th>密码</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Tanmay</td>
         <td>Bangalore</td>
         <td>560001</td>
      </tr>
      <tr>
         <td>Sachin</td>
         <td>Mumbai</td>
         <td>400003</td>
      </tr>
      <tr>
         <td>Uma</td>
         <td>Pune</td>
         <td>411027</td>
      </tr>
   </tbody>
</table>

</body>
</html>
```

结果如下所示：

条纹表格布局

名称	城市	密码
Tanmay	Bangalore	560001
Sachin	Mumbai	400003
Uma	Pune	411027

边框表格

通过添加 *.table-bordered* class，您将看到每个元素周围都有边框，且占整个表格是圆角的，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 边框表格</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<table class="table table-bordered">
   <caption>边框表格布局</caption>
   <thead>
      <tr>
         <th>名称</th>
         <th>城市</th>
         <th>密码</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Tanmay</td>
         <td>Bangalore</td>
         <td>560001</td>
      </tr>
      <tr>
         <td>Sachin</td>
         <td>Mumbai</td>
         <td>400003</td>
      </tr>
      <tr>
         <td>Uma</td>
         <td>Pune</td>
         <td>411027</td>
      </tr>
   </tbody>
</table>

</body>
</html>
```

结果如下所示：

边框表格布局

名称	城市	密码
Tanmay	Bangalore	560001
Sachin	Mumbai	400003
Uma	Pune	411027

悬停表格

通过添加 *.table-hover* class，当指针悬停在行上时会出现浅灰色背景，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 悬停表格</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<table class="table table-hover">
   <caption>悬停表格布局</caption>
   <thead>
      <tr>
         <th>名称</th>
         <th>城市</th>
         <th>密码</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Tanmay</td>
         <td>Bangalore</td>
         <td>560001</td>
      </tr>
      <tr>
         <td>Sachin</td>
         <td>Mumbai</td>
         <td>400003</td>
      </tr>
      <tr>
         <td>Uma</td>
         <td>Pune</td>
         <td>411027</td>
      </tr>
   </tbody>
</table>

</body>
</html>
```

结果如下所示：

悬停表格布局

名称	城市	密码
Tanmay	Bangalore	560001
Sachin	Mumbai	400003
Uma	Pune	411027

精简表格

通过添加 *.table-condensed* class，行内边距（padding）被切为两半，以便让表看起来更紧凑，如下面的实例所示。这在想让信息看起来更紧凑时非常有用。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 精简表格</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<table class="table table-condensed">
   <caption>精简表格布局</caption>
   <thead>
      <tr>
         <th>名称</th>
         <th>城市</th>
         <th>密码</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Tanmay</td>
         <td>Bangalore</td>
         <td>560001</td>
      </tr>
      <tr>
         <td>Sachin</td>
         <td>Mumbai</td>
         <td>400003</td>
      </tr>
      <tr>
         <td>Uma</td>
         <td>Pune</td>
         <td>411027</td>
      </tr>
   </tbody>
</table>

</body>
</html>
```

结果如下所示：

精简表格布局

名称	城市	密码
Tanmay	Bangalore	560001
Sachin	Mumbai	400003
Uma	Pune	411027

上下文类

下表中所列出的上下文类允许您改变表格行或单个单元格的背景颜色。

类	描述
.active	对某一特定的行或单元格应用悬停颜色
.success	表示一个成功的或积极的动作
.warning	表示一个需要注意的警告
.danger	表示一个危险的或潜在的负面动作

这些类可被应用到 <tr>、<td> 或 <th>。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 上下文类</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<table class="table">
   <caption>上下文表格布局</caption>
   <thead>
      <tr>
         <th>产品</th>
         <th>付款日期</th>
         <th>状态</th>
      </tr>
   </thead>
   <tbody>
      <tr class="active">
         <td>产品1</td>
         <td>23/11/2013</td>
         <td>待发货</td>
      </tr>
      <tr class="success">
         <td>产品2</td>
         <td>10/11/2013</td>
         <td>发货中</td>
      </tr>
      <tr  class="warning">
         <td>产品3</td>
         <td>20/10/2013</td>
         <td>待确认</td>
      </tr>
      <tr  class="danger">
         <td>产品4</td>
         <td>20/10/2013</td>
         <td>已退货</td>
      </tr>
   </tbody>
</table>

</body>
</html>
```

结果如下所示：

上下文表格布局

产品	付款日期	状态
产品1	23/11/2013	待发货
产品2	10/11/2013	发货中
产品3	20/10/2013	待确认
产品4	20/10/2013	已退货

响应式表格

通过把任意的 *.table* 包在 *.table-responsive* class 内，您可以让表格水平滚动以适应小型设备（小于 768px）。当在大于 768px 宽的大型设备上查看时，您将看不到任何的差别。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 响应式表格</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="table-responsive">
   <table class="table">
      <caption>响应式表格布局</caption>
      <thead>
         <tr>
            <th>产品</th>
            <th>付款日期</th>
            <th>状态</th>
         </tr>
      </thead>
      <tbody>
         <tr>
            <td>产品1</td>
            <td>23/11/2013</td>
            <td>待发货</td>
         </tr>
         <tr>
            <td>产品2</td>
            <td>10/11/2013</td>
            <td>发货中</td>
         </tr>
         <tr>
            <td>产品3</td>
            <td>20/10/2013</td>
            <td>待确认</td>
         </tr>
         <tr>
            <td>产品4</td>
            <td>20/10/2013</td>
            <td>已退货</td>
         </tr>
      </tbody>
   </table>
</div>

</body>
</html>
```

结果如下所示：

响应式表格布局

产品	付款日期	状态
产品1	23/11/2013	待发货
产品2	10/11/2013	发货中
产品3	20/10/2013	待确认
产品4	20/10/2013	已退货

Bootstrap 表单

在本章中，我们将学习如何使用 Bootstrap 创建表单。Bootstrap 通过一些简单的 HTML 标记和扩展的类即可创建出不同样式的表单。

表单布局

Bootstrap 提供了下列类型的表单布局：

- 垂直表单（默认）
- 内联表单
- 水平表单

垂直或基本表单

基本的表单结构是 Bootstrap 自带的，个别的表单控件自动接收一些全局样式。下面列出了创建基本表单的步骤：

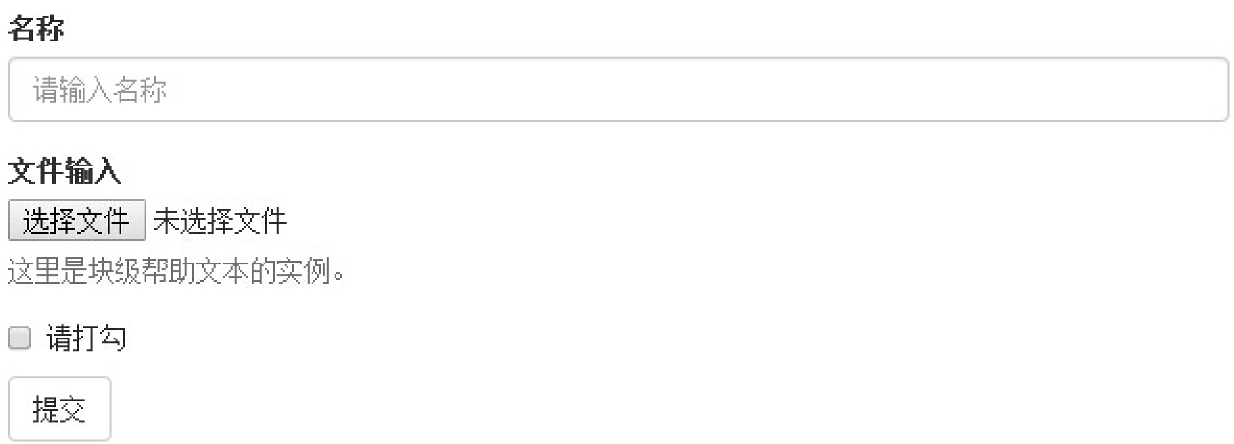
- 向父 <form> 元素添加 *role="form"*。
- 把标签和控件放在一个带有 class *.form-group* 的 <div> 中。这是获取最佳间距所必需的。
- 向所有的文本元素 <input>、<textarea> 和 <select> 添加 class *.form-control*。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本表单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form role="form">
   <div class="form-group">
      <label for="name">名称</label>
      <input type="text" class="form-control" id="name"
         placeholder="请输入名称">
   </div>
   <div class="form-group">
      <label for="inputfile">文件输入</label>
      <input type="file" id="inputfile">
      <p class="help-block">这里是块级帮助文本的实例。</p>
   </div>
   <div class="checkbox">
      <label>
      <input type="checkbox"> 请打勾
      </label>
   </div>
   <button type="submit" class="btn btn-default">提交</button>
</form>

</body>
</html>
```

结果如下所示：



内联表单

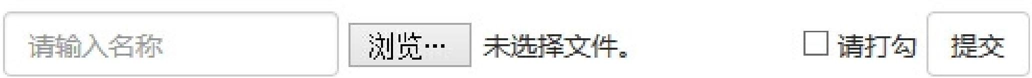
如果需要创建一个表单，它的所有元素是内联的，向左对齐的，标签是并排的，请向 <form> 标签添加 class *.form-inline*。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 内联表单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form class="form-inline" role="form">
   <div class="form-group">
      <label class="sr-only" for="name">名称</label>
      <input type="text" class="form-control" id="name"
         placeholder="请输入名称">
   </div>
   <div class="form-group">
      <label class="sr-only" for="inputfile">文件输入</label>
      <input type="file" id="inputfile">
   </div>
   <div class="checkbox">
      <label>
      <input type="checkbox"> 请打勾
      </label>
   </div>
   <button type="submit" class="btn btn-default">提交</button>
</form>

</body>
</html>
```

结果如下所示：



- 默认情况下，Bootstrap 中的 input、select 和 textarea 有 100% 宽度。在使用内联表单时，您需要在表单控件上设置一个宽度。
- 使用 class *.sr-only*，您可以隐藏内联表单的标签。

水平表单

水平表单与其他表单不仅标记的数量上不同，而且表单的呈现形式也不同。如需创建一个水平布局的表单，请按下面的几个步骤进行：

- 向父 <form> 元素添加 class *.form-horizontal*。

- 把标签和控件放在一个带有 class *.form-group* 的 <div> 中。
- 向标签添加 class *.control-label*。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 水平表单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form class="form-horizontal" role="form">
   <div class="form-group">
      <label for="firstname" class="col-sm-2 control-label">名字</l
      <div class="col-sm-10">
         <input type="text" class="form-control" id="firstname"
            placeholder="请输入名字">
      </div>
   </div>
   <div class="form-group">
      <label for="lastname" class="col-sm-2 control-label">姓</labe
      <div class="col-sm-10">
         <input type="text" class="form-control" id="lastname"
            placeholder="请输入姓">
      </div>
   </div>
   <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
         <div class="checkbox">
            <label>
               <input type="checkbox"> 请记住我
            </label>
         </div>
      </div>
   </div>
   <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
         <button type="submit" class="btn btn-default">登录</button
      </div>
   </div>
</form>

</body>
</html>
```

结果如下所示：



支持的表单控件

Bootstrap 支持最常见的表单控件，主要是 *input*、*textarea*、*checkbox*、*radio* 和 *select*。

输入框（Input）

最常见的表单文本字段是输入框 input。用户可以在其中输入大多数必要的表单数据。Bootstrap 提供了对所有原生的 HTML5 的 input 类型的支持，包括：*text*、*password*、*datetime*、*datetime-local*、*date*、*month*、*time*、*week*、*number*、*email*、*url*、*search*、*tel* 和 *color*。适当的 *type* 声明是必需的，这样才能让 *input* 获得完整的样式。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 输入框</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form role="form">
  <div class="form-group">
    <label for="name">标签</label>
    <input type="text" class="form-control" placeholder="文本输入">
  </div>
 </form>

</body>
</html>
```

结果如下所示：

标签

文本输入

文本框（Textarea）

当您需要进行多行输入的时，则可以使用文本框 textarea。必要时可以改变 *rows* 属性（较少的行 = 较小的盒子，较多的行 = 较大的盒子）。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 文本框</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form role="form">
  <div class="form-group">
    <label for="name">文本框</label>
    <textarea class="form-control" rows="3"></textarea>
  </div>
</form>

</body>
</html>
```

结果如下所示：

文本框

复选框（CheckBoxe）和单选框（Radio）

复选框和单选按钮用于让用户从一系列预设置的选项中进行选择。

- 当创建表单时，如果您想让用户从列表中选择若干个选项时，请使用 *checkbox*。如果您限制用户只能选择一个选项，请使用 *radio*。
- 对一系列复选框和单选框使用 *.checkbox-inline* 或 *.radio-inline* class，控制它们显示在同一行上。

下面的实例演示了这两种类型（默认和内联）：

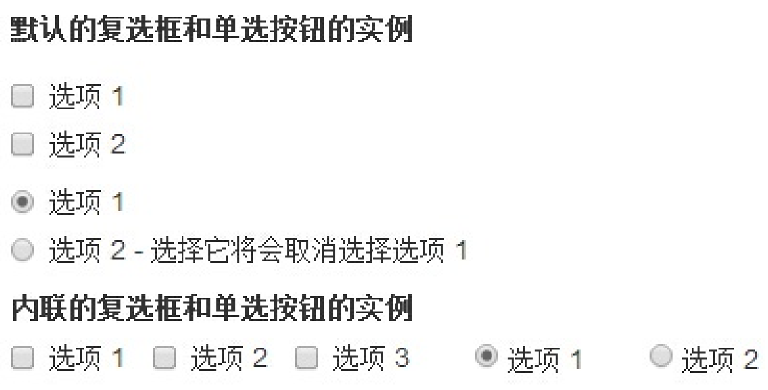
```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 复选框和单选按钮</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<label for="name">默认的复选框和单选按钮的实例</label>
<div class="checkbox">
   <label><input type="checkbox" value="">选项 1</label>
</div>
<div class="checkbox">
   <label><input type="checkbox" value="">选项 2</label>
</div>

<div class="radio">
   <label>
      <input type="radio" name="optionsRadios" id="optionsRadios1"
         value="option1" checked> 选项 1
   </label>
</div>
<div class="radio">
   <label>
      <input type="radio" name="optionsRadios" id="optionsRadios2"
         value="option2">
         选项 2 - 选择它将会取消选择选项 1
   </label>
</div>
<label for="name">内联的复选框和单选按钮的实例</label>
<div>
   <label class="checkbox-inline">
      <input type="checkbox" id="inlineCheckbox1" value="option1">
   </label>
   <label class="checkbox-inline">
      <input type="checkbox" id="inlineCheckbox2" value="option2">
   </label>
   <label class="checkbox-inline">
      <input type="checkbox" id="inlineCheckbox3" value="option3">
   </label>
   <label class="checkbox-inline">
      <input type="radio" name="optionsRadiosinline" id="optionsRad
         value="option1" checked> 选项 1
   </label>
   <label class="checkbox-inline">
      <input type="radio" name="optionsRadiosinline" id="optionsRad
         value="option2"> 选项 2
   </label>
</div>
```

```
</body>
</html>
```

结果如下所示：



选择框（Select）

当您想让用户从多个选项中进行选择，但是默认情况下只能选择一个选项时，则使用选择框。

- 使用 <select> 展示列表选项，通常是那些用户很熟悉的选择列表，比如州或者数字。
- 使用 *multiple="multiple"* 允许用户选择多个选项。

下面的实例演示了这两种类型（select 和 multiple）：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 选择框</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form role="form">
   <div class="form-group">
      <label for="name">选择列表</label>
      <select class="form-control">
         <option>1</option>
         <option>2</option>
         <option>3</option>
         <option>4</option>
         <option>5</option>
      </select>

      <label for="name">可多选的选择列表</label>
      <select multiple class="form-control">
         <option>1</option>
         <option>2</option>
         <option>3</option>
         <option>4</option>
         <option>5</option>
      </select>
   </div>
</form>

</body>
</html>
```

结果如下所示：



静态控件

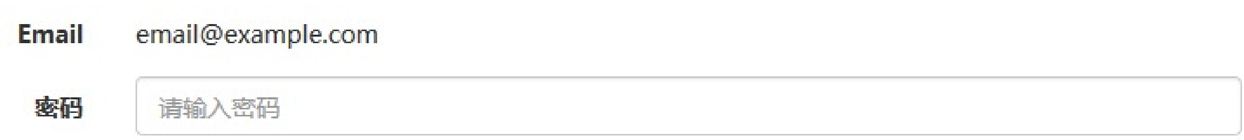
当您需要在一个水平表单内的表单标签后放置纯文本时，请在 <p> 上使用 class *.form-control-static*。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 静态控件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form class="form-horizontal" role="form">
  <div class="form-group">
    <label class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
      <p class="form-control-static">email@example.com</p>
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword" class="col-sm-2 control-label">密码<
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPasswor
         placeholder="请输入密码">
    </div>
  </div>
</form>

</body>
</html>
```

结果如下所示：



表单控件状态

除了 *:focus* 状态（即，用户点击 input 或使用 tab 键聚焦到 input 上），Bootstrap 还为禁用的输入框定义了样式，并提供了表单验证的 class。

输入框焦点

当输入框 input 接收到 *:focus* 时，输入框的轮廓会被移除，同时应用 *box-shadow*。

禁用的输入框 input

如果您想要禁用一个输入框 input，只需要简单地添加 *disabled* 属性，这不仅会禁用输入框，还会改变输入框的样式以及当鼠标的指针悬停在元素上时鼠标指针的样式。

禁用的字段集 fieldset

对 <fieldset> 添加 disabled 属性来禁用 <fieldset> 内的所有控件。

验证状态

Bootstrap 包含了错误、警告和成功消息的验证样式。只需要对父元素简单地添加适当的 class（*.has-warning*、 *.has-error* 或 *.has-success*）即可使用验证状态。

下面的实例演示了所有控件状态：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 表单控件状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form class="form-horizontal" role="form">
   <div class="form-group">
      <label class="col-sm-2 control-label">聚焦</label>
      <div class="col-sm-10">
         <input class="form-control" id="focusedInput" type="text"
            value="该输入框获得焦点...">
      </div>
   </div>
   <div class="form-group">
      <label for="inputPassword" class="col-sm-2 control-label">
         禁用
      </label>
      <div class="col-sm-10">
         <input class="form-control" id="disabledInput" type="text"
            placeholder="该输入框禁止输入..." disabled>
      </div>
   </div>
   <fieldset disabled>
      <div class="form-group">
```

```
            <label for="disabledTextInput"  class="col-sm-2 control-la
               禁用输入（Fieldset disabled）
            </label>
            <div class="col-sm-10">
               <input type="text" id="disabledTextInput" class="form-c
                  placeholder="禁止输入">
            </div>
         </div>
         <div class="form-group">
            <label for="disabledSelect"  class="col-sm-2 control-label
               禁用选择菜单（Fieldset disabled）
            </label>
            <div class="col-sm-10">
               <select id="disabledSelect" class="form-control">
                  <option>禁止选择</option>
               </select>
            </div>
         </div>
      </fieldset>
      <div class="form-group has-success">
         <label class="col-sm-2 control-label" for="inputSuccess">
            输入成功
         </label>
         <div class="col-sm-10">
            <input type="text" class="form-control" id="inputSuccess">
         </div>
      </div>
      <div class="form-group has-warning">
         <label class="col-sm-2 control-label" for="inputWarning">
            输入警告
         </label>
         <div class="col-sm-10">
            <input type="text" class="form-control" id="inputWarning">
         </div>
      </div>
      <div class="form-group has-error">
         <label class="col-sm-2 control-label" for="inputError">
            输入错误
         </label>
         <div class="col-sm-10">
            <input type="text" class="form-control" id="inputError">
         </div>
      </div>
   </form>

   </body>
   </html>
```

结果如下所示：



表单控件大小

您可以分别使用 class *.input-lg* 和 *.col-lg-** 来设置表单的高度和宽度。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 表单控件大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form role="form">
   <div class="form-group">
      <input class="form-control input-lg" type="text"
         placeholder=".input-lg">
   </div>

   <div class="form-group">
      <input class="form-control" type="text" placeholder="默认输入"
   </div>

   <div class="form-group">
      <input class="form-control input-sm" type="text"
         placeholder=".input-sm">
   </div>
   <div class="form-group">
   </div>
```

```
    <div class="form-group">
       <select class="form-control input-lg">
          <option value="">.input-lg</option>
       </select>
    </div>
    <div class="form-group">
       <select class="form-control">
          <option value="">默认选择</option>
       </select>
    </div>
    <div class="form-group">
       <select class="form-control input-sm">
          <option value="">.input-sm</option>
       </select>
    </div>

    <div class="row">
       <div class="col-lg-2">
          <input type="text" class="form-control" placeholder=".col-
       </div>
       <div class="col-lg-3">
          <input type="text" class="form-control" placeholder=".col-
       </div>
       <div class="col-lg-4">
          <input type="text" class="form-control" placeholder=".col-
       </div>
    </div>
 </form>

 </body>
 </html>
```

结果如下所示：



表单帮助文本

Bootstrap 表单控件可以在输入框 input 上有一个块级帮助文本。为了添加一个占用整个宽度的内容块，请在 <input> 后使用 *.help-block*。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 表单帮助文本</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<form role="form">
   <span>帮助文本实例</span>
   <input class="form-control" type="text" placeholder="">
   <span class="help-block">一个较长的帮助文本块，超过一行，
   需要扩展到下一行。本实例中的帮助文本总共有两行。</span>
</form>

</body>
</html>
```

结果如下所示：

帮助文本实例

一个较长的帮助文本块，超过一行，需要扩展到下一行。本实例中的帮助文本总共有两行。

Bootstrap 按钮

本章将通过实例讲解如何使用 Bootstrap 按钮。任何带有 class **.btn** 的元素都会继承圆角灰色按钮的默认外观。但是 Bootstrap 提供了一些选项来定义按钮的样式，具体如下表所示：

Class	描述
btn	默认的/标准的按钮。
btn-primary	提供额外的视觉效果，标识一组按钮中的原始动作。
btn-success	表示一个成功的或积极的动作。
btn-info	信息警告消息的上下文按钮。
btn-warning	表示应谨慎采取的动作。
btn-danger	表示一个危险的或潜在的负面动作。
btn-link	并不强调是一个按钮，看起来像一个链接，但同时保持按钮的行为。

下面的实例演示了上面所有的按钮 class：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮选项</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<!-- 标准的按钮 -->
<button type="button" class="btn btn-default">默认按钮</button>

<!-- 提供额外的视觉效果，标识一组按钮中的原始动作 -->
<button type="button" class="btn btn-primary">原始按钮</button>

<!-- 表示一个成功的或积极的动作 -->
<button type="button" class="btn btn-success">成功按钮</button>

<!-- 信息警告消息的上下文按钮 -->
<button type="button" class="btn btn-info">信息按钮</button>

<!-- 表示应谨慎采取的动作 -->
<button type="button" class="btn btn-warning">警告按钮</button>

<!-- 表示一个危险的或潜在的负面动作 -->
<button type="button" class="btn btn-danger">危险按钮</button>

<!-- 并不强调是一个按钮，看起来像一个链接，但同时保持按钮的行为 -->
<button type="button" class="btn btn-link">链接按钮</button>

</body>
</html>
```

结果如下所示：



按钮大小

下表列出了获得各种大小按钮的 class：

Class	描述
.btn-lg	这会让按钮看起来比较大。
.btn-sm	这会让按钮看起来比较小。
.btn-xs	这会让按钮看起来特别小。
.btn-block	这会创建块级的按钮，会横跨父元素的全部宽度。

下面的实例演示了上面所有的按钮 class：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>
   <button type="button" class="btn btn-primary btn-lg">
      大的原始按钮
   </button>
   <button type="button" class="btn btn-default btn-lg">
      大的按钮
   </button>
</p>
<p>
   <button type="button" class="btn btn-primary">
      默认大小的原始按钮
   </button>
   <button type="button" class="btn btn-default">
      默认大小的按钮
   </button>
</p>
<p>
   <button type="button" class="btn btn-primary btn-sm">
      小的原始按钮
   </button>
   <button type="button" class="btn btn-default btn-sm">
      小的按钮
   </button>
</p>
<p>
   <button type="button" class="btn btn-primary btn-xs">
      特别小的原始按钮
   </button>
   <button type="button" class="btn btn-default btn-xs">
      特别小的按钮
```

```
        </button>
    </p>
    <p>
        <button type="button" class="btn btn-primary btn-lg btn-block">
            块级的原始按钮
        </button>
        <button type="button" class="btn btn-default btn-lg btn-block">
            块级的按钮
        </button>
    </p>

    </body>
    </html>
```

结果如下所示：



按钮状态

Bootstrap 提供了激活、禁用等按钮状态的 class，下面将进行详细讲解。

激活状态

按钮在激活时将呈现为被按压的外观（深色的背景、深色的边框、阴影）。

下表列出了让按钮元素和锚元素呈激活状态的 class：

元素	Class
按钮元素	添加 **.active** class 来显示它是激活的。
锚元素	添加 **.active** class 到 <a> 按钮来显示它是激活的。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮激活状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>
   <button type="button" class="btn btn-default btn-lg ">
      默认按钮
   </button>
   <button type="button" class="btn btn-default btn-lg active">
      激活按钮
   </button>
</p>
<p>
   <button type="button" class="btn btn-primary btn-lg ">
      原始按钮
   </button>
   <button type="button" class="btn btn-primary btn-lg active">
      激活的原始按钮
   </button>
</p>

</body>
</html>
```

结果如下所示：



禁用状态

当您禁用一个按钮时，它的颜色会变淡 50%，并失去渐变。

下表列出了让按钮元素和锚元素呈禁用状态的 class：

元素	Class
按钮元素	添加 **disabled** 属性 到 <button> 按钮。
锚元素	添加 **disabled** *class* 到 <a> 按钮。 注意：该 *class* 只会改变 <*a*> 的外观，不会改变它的功能。在这里，您需要使用自定义的 *JavaScript* 来禁用链接。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮禁用状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>
   <button type="button" class="btn btn-default btn-lg">
      默认按钮
   </button>
   <button type="button" class="btn btn-default btn-lg" disabled="c
      禁用按钮
   </button>
</p>
<p>
   <button type="button" class="btn btn-primary btn-lg ">
      原始按钮
   </button>
   <button type="button" class="btn btn-primary btn-lg" disabled="c
      禁用的原始按钮
   </button>
</p>
<p>
   <a href="#" class="btn btn-default btn-lg" role="button">
      链接
   </a>
   <a href="#" class="btn btn-default btn-lg disabled" role="button
      禁用链接
   </a>
</p>
<p>
   <a href="#" class="btn btn-primary btn-lg" role="button">
      原始链接
   </a>
   <a href="#" class="btn btn-primary btn-lg disabled" role="button
      禁用的原始链接
   </a>
</p>

</body>
</html>
```

结果如下所示：



按钮标签

您可以在 <a>、<button> 或 <input> 元素上使用按钮 class。但是建议您在 <button> 元素上使用按钮 class，避免跨浏览器的不一致性问题。

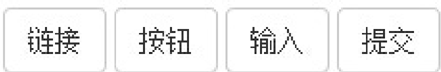
下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮标签</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<a class="btn btn-default" href="#" role="button">链接</a>
<button class="btn btn-default" type="submit">按钮</button>
<input class="btn btn-default" type="button" value="输入">
<input class="btn btn-default" type="submit" value="提交">

</body>
</html>
```

结果如下所示：



Bootstrap 图像

在本章中，我们将学习 Bootstrap 对图像的支持。Bootstrap 提供了三个可对图像应用简单样式的 class：

- *.img-rounded*：添加 *border-radius:6px* 来获得图像圆角。
- *.img-circle*：添加 *border-radius:500px* 来让整个图像变成圆形。
- *.img-thumbnail*：添加一些内边距（padding）和一个灰色的边框。

请看下面的实例演示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 图像</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<img src="img/download.png"
   class="img-rounded">
<img src="img/download.png"
   class="img-circle">
<img src="img/download.png"
   class="img-thumbnail">

</body>
</html>
```

结果如下所示：



Bootstrap 帮助器类

本章将讨论 Bootstrap 中的一些可能会派上用场的帮助器类。

关闭图标

使用通用的关闭图标来关闭模态框和警告框。使用 class **close** 得到关闭图标。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 关闭图标</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>关闭图标实例
   <button type="button" class="close" aria-hidden="true">
      &times;
   </button>
</p>

</body>
</html>
```

结果如下所示：



插入符

使用插入符表示下拉功能和方向。使用带有 class **caret** 的 <span> 元素得到该功能。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 插入符</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>插入符实例
   <span class="caret"></span>
</p>

</body>
</html>
```

结果如下所示：

插入符实例 ▾

快速浮动

您可以分别使用 class **pull-left** 或 **pull-right** 来把元素向左或向右浮动。下面的实例演示了这点。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 快速浮动</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="pull-left">
   向左快速浮动
</div>
<div class="pull-right">
   向右快速浮动
</div>

</body>
</html>
```

结果如下所示：

Quick Float to left　　　　　　Quick Float to right

如需对齐导航栏中的组件，请使用 **.navbar-left** 或 **.navbar-right** 代替。请查看 Bootstrap 导航栏。

居中内容块

使用 class **center-block** 来居中元素。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 居中内容块</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="row">
   <div class="center-block" style="width:200px;background-color:#c
      这是 center-block 实例
   </div>
</div>

</body>
</html>
```

结果如下所示：

这是 center-block 实例

清除浮动

如需清除元素的浮动，请使用 **.clearfix** class。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 清除浮动</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="clearfix"  style="background: #D8D8D8;border: 1px soli
   <div class="pull-left" style="background:#58D3F7;">
      向左快速浮动
   </div>
   <div class="pull-right" style="background: #DA81F5;">
      向右快速浮动
   </div>
</div>

</body>
</html>
```

结果如下所示：



显示和隐藏内容

您可以通过使用 class **.show** 和 **.hidden** 来强行设置元素显示或隐藏（包括屏幕阅读器）。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 显示和隐藏内容</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="row" style="padding: 91px 100px 19px 50px;">
   <div class="show" style="left-margin:10px;width:300px;backgroun
      这是 show class 的实例
   </div>
   <div class="hidden" style="width:200px;background-color:#ccc;">
      这是 hide class 的实例
   </div>
</div>

</body>
</html>
```

结果如下所示：



屏幕阅读器

您可以通过使用 class **.sr-only** 来把元素对所有设备隐藏，除了屏幕阅读器。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 屏幕阅读器</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="row" style="padding: 91px 100px 19px 50px;">
   <form class="form-inline" role="form">
   <div class="form-group">
      <label class="sr-only" for="email">Email 地址</label>
      <input type="email" class="form-control" placeholder="Enter e
   </div>
   <div class="form-group">
      <label class="sr-only" for="pass">密码</label>
      <input type="password" class="form-control" placeholder="Pass
   </div>
</div>

</body>
</html>
```

结果如下所示：



在这里，我们看到两个 input 类型的 label 标签都带有 class **sr-only**，因此标签将只对屏幕阅读器可见。

Bootstrap 响应式实用工具

Bootstrap 提供了一些帮助器类，以便更快地实现对移动设备友好的开发。这些可以通过媒体查询结合大型、小型和中型设备，实现内容对设备的显示和隐藏。

需要谨慎使用这些工具，避免在同一个站点创建完全不同的版本。响应式实用工具目前只适用于块和表切换。

Class	设备
.visible-xs	额外的小设备（小于 768px）可见
.visible-sm	小型设备（768 px 起）可见
.visible-md	中型设备（768 px 到 991 px）可见
.visible-lg	大型设备（992 px 及以上）可见
.hidden-xs	额外的小设备（小于 768px）隐藏
.hidden-sm	小型设备（768 px 起）隐藏
.hidden-md	中型设备（768 px 到 991 px）隐藏
.hidden-lg	大型设备（992 px 及以上）隐藏

打印类

下表列出了打印类。使用这些切换打印内容。

Class	打印
.visible-print	可见，可打印
.hidden-print	只对浏览器可见，不可打印

实例

下面的实例演示了上面所列举的帮助器类的用法。调整浏览器的窗口大小，或者在不同的设备上加载实例，测试响应式实用工具类。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 响应式实用工具</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container" style="padding: 40px;">
   <div class="row visible-on">
      <div class="col-xs-6 col-sm-3" style="background-color: #dede
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         <span class="hidden-xs">特别小型</span>
         <span class="visible-xs">? 在特别小型设备上可见</span>
      </div>
      <div class="col-xs-6 col-sm-3" style="background-color: #dede
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         <span class="hidden-sm">小型</span>
         <span class="visible-sm">? 在小型设备上可见</span>
      </div>
      <div class="clearfix visible-xs"></div>
      <div class="col-xs-6 col-sm-3" style="background-color: #dede
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         <span class="hidden-md">中型</span>
         <span class="visible-md">? 在中型设备上可见</span>
      </div>
      <div class="col-xs-6 col-sm-3" style="background-color: #dede
         box-shadow: inset 1px -1px 1px #444, inset -1px 1px 1px #4
         <span class="hidden-lg">大型</span>
         <span class="visible-lg">? 在大型设备上可见</span>
      </div>
</div>

</body>
</html>
```

结果如下所示：



勾号（?） 表示元素在当前视口中可见。

Bootstrap 字形图标（Glyphicons）

本章将讲解字形图标（Glyphicons），并通过一些实例了解它的使用。Bootstrap 捆绑了 200 多种字体格式的自行。首先让我们先来理解一下什么是字形图标（Glyphicons）。

什么是字形图标（Glyphicons）？

字形图标（Glyphicons）是在 Web 项目中使用的图标字体。虽然，Glyphicons Halflings 需要商业许可，但是您可以通过基于项目的 Bootstrap 来免费使用这些图标。

为了表示对图标作者的感谢，希望您在使用时加上 GLYPHICONS 网站的链接。

获取字形图标（Glyphicons）

我们已经在 环境安装 章节下载了 Bootstrap 3.x 版本，并理解了它的目录结构。在 *fonts* 文件夹内可以找到字形图标（Glyphicons），它包含了下列这些文件：

- glyphicons-halflings-regular.eot
- glyphicons-halflings-regular.svg
- glyphicons-halflings-regular.ttf
- glyphicons-halflings-regular.woff

相关的 CSS 规则写在 *dist* 文件夹内的 css 文件夹内的 *bootstrap.css* 和 *bootstrap-min.css* 文件上。

字形图标（Glyphicons）列表

点击这里，查看可用的字形图标（Glyphicons）列表。

CSS 规则解释

下面的 CSS 规则构成 glyphicon class。

```
@font-face {
  font-family: 'Glyphicons Halflings';
  src: url('../fonts/glyphicons-halflings-regular.eot');
  src: url('../fonts/glyphicons-halflings-regular.eot?#iefix') forr
}

.glyphicon {
  position: relative;
  top: 1px;
  display: inline-block;
  font-family: 'Glyphicons Halflings';
  -webkit-font-smoothing: antialiased;
  font-style: normal;
  font-weight: normal;
  line-height: 1;
  -moz-osx-font-smoothing: grayscale;
}
```

所以 font-face 规则实际上是在找到 glyphicons 地方声明 font-family 和位置。

.glyphicon class 声明一个从顶部偏移 1px 的相对位置，呈现为 inline-block，声明字体，规定 font-style 和 font-weight 为 normal，设置行高为 1。除此之外，使用 *-webkit-font-smoothing: antialiased* 和 *-moz-osx-font-smoothing: grayscale;* 获得跨浏览器的一致性。

然后，这里的

```
.glyphicon:empty {
  width: 1em;
}
```

是空的 glyphicon。

这里有 200 个 class，每个 class 针对一个图标。这些 class 的常见格式如下：

```
.glyphicon-keyword:before {
  content: "hexvalue";
}
```

比如，使用的 user 图标，它的 class 如下：

```
.glyphicon-user:before {
  content: "\e008";
}
```

用法

如需使用图标，只需要简单地使用下面的代码即可。请在图标和文本之间保留适当的空间。

```
<span class="glyphicon glyphicon-search"></span>
```

下面的实例演示了如何使用字形图标（Glyphicons）：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 如何使用字形图标（Glyphicons）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>
   <button type="button" class="btn btn-default">
      <span class="glyphicon glyphicon-sort-by-attributes"></span>
   </button>
   <button type="button" class="btn btn-default">
      <span class="glyphicon glyphicon-sort-by-attributes-alt"></sp
   </button>
   <button type="button" class="btn btn-default">
      <span class="glyphicon glyphicon-sort-by-order"></span>
   </button>
   <button type="button" class="btn btn-default">
      <span class="glyphicon glyphicon-sort-by-order-alt"></span>
   </button>
</p>
<button type="button" class="btn btn-default btn-lg">
  <span class="glyphicon glyphicon-user"></span> User
</button>
<button type="button" class="btn btn-default btn-sm">
  <span class="glyphicon glyphicon-user"></span> User
</button>
<button type="button" class="btn btn-default btn-xs">
  <span class="glyphicon glyphicon-user"></span> User
</button>

</body>
</html>
```

结果如下所示：



带有导航栏的字形图标（Glyphicons）

```
<!DOCTYPE html>
<html>
  <head>
    <title>导航栏的字形图标</title>
    <meta name="viewport" content="width=device-width, initial-scal
    <!-- Bootstrap -->
    <link href="http://apps.bdimg.com/libs/bootstrap/3.2.0/css/boot
    <style>
    body {
    padding-top: 50px;
    padding-left: 50px;
    }
    </style>
    <!--[if lt IE 9]>
      <script src="http://apps.bdimg.com/libs/html5shiv/3.7/html5sh
    <![endif]-->
  </head>
  <body>
    <div class="navbar navbar-fixed-top navbar-inverse" role="navig
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle" data-toggle='
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Project name</a>
        </div>
        <div class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="#"><span class="glyphicon g
            <li><a href="#shop"><span class="glyphicon glyphicon-sh
            <li><a href="#support"><span class="glyphicon glyphicon
          </ul>
        </div><!-- /.nav-collapse -->
      </div><!-- /.container -->
    </div>
    <!-- jQuery (Bootstrap 插件需要引入) -->
    <script src="http://apps.bdimg.com/libs/jquery/2.1.1/jquery.mir
    <!-- 包含了所有编译插件 -->
    <script src="http://apps.bdimg.com/libs/bootstrap/3.2.0/js/boot
  </body>
</html>
```

定制字形图标（Glyphicons）

我们已经看到如何使用字形图标（Glyphicons），接下来我们看看如何定制字形图标（Glyphicons）。

我们将以上面的实例开始，并通过改变字体尺寸、颜色和应用文本阴影来进行定制图标。

下面是开始的代码：

```
<button type="button" class="btn btn-primary btn-lg">
  <span class="glyphicon glyphicon-user"></span> User
</button>
```

效果如下所示：



定制字体尺寸

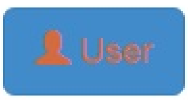
通过增加或减少图标的字体尺寸，您可以让图标看起来更大或更小。

```
<button type="button" class="btn btn-primary btn-lg" style="font-s
  <span class="glyphicon glyphicon-user"></span> User
</button>
```



定制字体颜色

```
<button type="button" class="btn btn-primary btn-lg" style="color:
  <span class="glyphicon glyphicon-user"></span> User
</button>
```



应用文本阴影

```
<button type="button" class="btn btn-primary btn-lg" style="text-sh
  <span class="glyphicon glyphicon-user"></span> User
</button>
```



在线定制字形图标（Glyphicons）

点击这里，在线定制字形图标（Glyphicons）。

Bootstrap 下拉菜单（Dropdowns）

本章将重点介绍 Bootstrap 下拉菜单。下拉菜单是可切换的，是以列表格式显示链接的上下文菜单。这可以通过与 下拉菜单（Dropdown） JavaScript 插件 的互动来实现。

如需使用下列菜单，只需要在 class **.dropdown** 内加上下拉菜单即可。下面的实例演示了基本的下拉菜单：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 下拉菜单（Dropdowns）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="dropdown">
   <button type="button" class="btn dropdown-toggle" id="dropdownMe
      data-toggle="dropdown">
      主题
      <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownM
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">Java</a>
      </li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">数据挖掘</a>
      </li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">
            数据通信/网络
         </a>
      </li>
      <li role="presentation" class="divider"></li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">分离的链接</a>
      </li>
   </ul>
</div>

</body>
</html>
```

结果如下所示：



选项

对齐

通过向 **.dropdown-menu** 添加 class **.pull-right** 来向右对齐下拉菜单。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 向右对齐下拉菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="dropdown">
   <button type="button" class="btn dropdown-toggle" id="dropdownMe
      data-toggle="dropdown">主题
      <span class="caret"></span>
   </button>
   <ul class="dropdown-menu pull-right" role="menu"
      aria-labelledby="dropdownMenu1">
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">Java</a>
      </li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">数据挖掘</a>
      </li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">
            数据通信/网络
         </a>
      </li>
      <li role="presentation" class="divider"></li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">分离的链接</a>
      </li>
   </ul>
</div>

</body>
</html>
```

结果如下所示：



标题

您可以使用 class **dropdown-header** 向下拉菜单的标签区域添加标题。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 下拉菜单标题</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="dropdown">
   <button type="button" class="btn dropdown-toggle" id="dropdownMe
      data-toggle="dropdown">
      主题
      <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownM
      <li role="presentation" class="dropdown-header">下拉菜单标题</l
      <li role="presentation" >
         <a role="menuitem" tabindex="-1" href="#">Java</a>
      </li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">数据挖掘</a>
      </li>
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">
            数据通信/网络
         </a>
      </li>
      <li role="presentation" class="divider"></li>
      <li role="presentation" class="dropdown-header">下拉菜单标题</l
      <li role="presentation">
         <a role="menuitem" tabindex="-1" href="#">分离的链接</a>
      </li>
   </ul>
</div>

</body>
</html>
```

结果如下所示：


主题
下拉菜单标题
Java
数据挖掘
数据通信/网络
下拉菜单标题
分离的链接

Bootstrap 按钮组

按钮组允许多个按钮被堆叠在同一行上。当你想要把按钮对齐在一起时，这就显得非常有用。您可以通过 Bootstrap 按钮（Button） 插件 添加可选的 JavaScript 单选框和复选框样式行为。

下面的表格总结了 Bootstrap 提供的使用按钮组的一些重要的 class：

.btn-group

该 class 用于形成基本的按钮组。在 **.btn-group** 中放置一系列带有 class **.btn** 的按钮。

```
&lt;div class="btn-group"&gt;
  &lt;button type="button" class="btn btn-default"&gt;Button1&lt;/
   &lt;button type="button" class="btn btn-default"&gt;Button2&lt;
&lt;/div&gt;
```

.btn-toolbar

该 class 有助于把几组 <div class="btn-group"> 结合到一个 <div class="btn-toolbar"> 中，一般获得更复杂的组件。

```
&lt;div class="btn-toolbar" role="toolbar"&gt;
  &lt;div class="btn-group"&gt;...&lt;/div&gt;
  &lt;div class="btn-group"&gt;...&lt;/div&gt;
&lt;/div&gt;
```

.btn-group-lg, .btn-group-sm, .btn-group-xs

这些 class 可应用到整个按钮组的大小调整，而不需要对每个按钮进行大小调整。

```
&lt;div class="btn-group btn-group-lg"&gt;...&lt;/div&gt;
&lt;div class="btn-group btn-group-sm"&gt;...&lt;/div&gt;
&lt;div class="btn-group btn-group-xs"&gt;...&lt;/div&gt;
```

.btn-group-vertical

该 class 让一组按钮垂直堆叠显示，而不是水平堆叠显示。

```
&lt;div class="btn-group-vertical"&gt;
  ...
&lt;/div&gt;
```

基本的按钮组

下面的实例演示了上面表格中讨论到的 class **.btn-group** 的使用：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本的按钮组</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group">
  <button type="button" class="btn btn-default">按钮 1</button>
  <button type="button" class="btn btn-default">按钮 2</button>
  <button type="button" class="btn btn-default">按钮 3</button>
</div>

</body>
</html>
```

结果如下所示：



按钮工具栏

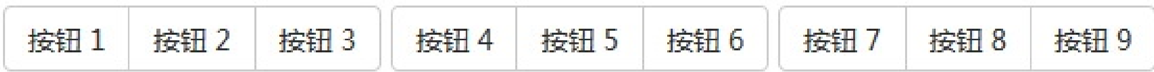
下面的实例演示了上面表格中讨论到的 class **.btn-toolbar** 的使用：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮工具栏</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-toolbar" role="toolbar">
  <div class="btn-group">
  <button type="button" class="btn btn-default">按钮 1</button>
  <button type="button" class="btn btn-default">按钮 2</button>
  <button type="button" class="btn btn-default">按钮 3</button>
 </div>
  <div class="btn-group">
  <button type="button" class="btn btn-default">按钮 4</button>
  <button type="button" class="btn btn-default">按钮 5</button>
  <button type="button" class="btn btn-default">按钮 6</button>
  </div>
  <div class="btn-group">
  <button type="button" class="btn btn-default">按钮 7</button>
  <button type="button" class="btn btn-default">按钮 8</button>
  <button type="button" class="btn btn-default">按钮 9</button>
  </div>
</div>

</body>
</html>
```

结果如下所示：



按钮的大小

下面的实例演示了上面表格中讨论到的 class **.btn-group-*** 的使用：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮组的大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group btn-group-lg">
  <button type="button" class="btn btn-default">按钮 1</button>
  <button type="button" class="btn btn-default">按钮 2</button>
  <button type="button" class="btn btn-default">按钮 3</button>
 </div>
  <div class="btn-group btn-group-sm">
  <button type="button" class="btn btn-default">按钮 4</button>
  <button type="button" class="btn btn-default">按钮 5</button>
  <button type="button" class="btn btn-default">按钮 6</button>
</div>
  <div class="btn-group btn-group-xs">
  <button type="button" class="btn btn-default">按钮 7</button>
  <button type="button" class="btn btn-default">按钮 8</button>
  <button type="button" class="btn btn-default">按钮 9</button>
</div>

</body>
</html>
```

结果如下所示：



嵌套

您可以在一个按钮组内嵌套另一个按钮组，即，在一个 **.btn-group** 内嵌套另一个 **.btn-group** 。当您向下拉菜单与一系列按钮组合使用时，就会用到这个。

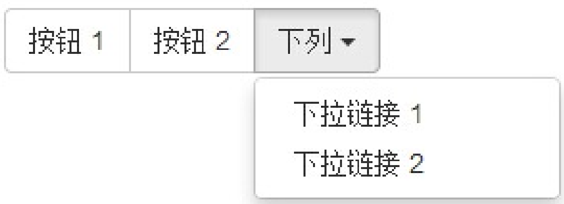
```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 嵌套的按钮组</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group">
  <button type="button" class="btn btn-default">按钮 1</button>
  <button type="button" class="btn btn-default">按钮 2</button>

  <div class="btn-group">
    <button type="button" class="btn btn-default dropdown-toggle"
      data-toggle="dropdown">
      下列
      <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
      <li><a href="#">下拉链接 1</a></li>
      <li><a href="#">下拉链接 2</a></li>
    </ul>
  </div>
</div>

</body>
</html>
```

结果如下所示：



垂直的按钮组

下面的实例演示了上面表格中讨论到的 class **.btn-group-vertical** 的使用：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 垂直的按钮组</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group-vertical">
  <button type="button" class="btn btn-default">按钮 1</button>
  <button type="button" class="btn btn-default">按钮 2</button>

  <div class="btn-group-vertical">
    <button type="button" class="btn btn-default dropdown-toggle"
      data-toggle="dropdown">
      下拉
      <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
      <li><a href="#">下拉链接 1</a></li>
      <li><a href="#">下拉链接 2</a></li>
    </ul>
  </div>
</div>

</body>
</html>
```

结果如下所示：



Bootstrap 按钮下拉菜单

本章将讲解如何使用 Bootstrap class 向按钮添加下拉菜单。如需向按钮添加下拉菜单，只需要简单地在在一个 **.btn-group** 中放置按钮和下拉菜单即可。您也可以使用 来指示按钮作为下拉菜单。

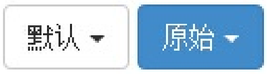
下面的实例演示了一个基本的简单的按钮下拉菜单：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本的按钮下拉菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group">
   <button type="button" class="btn btn-default dropdown-toggle"
      data-toggle="dropdown">
      默认 <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
   </ul>
</div>
<div class="btn-group">
   <button type="button" class="btn btn-primary dropdown-toggle"
      data-toggle="dropdown">
      原始 <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
   </ul>
</div>

</body>
</html>
```

结果如下所示：



分割的按钮下拉菜单

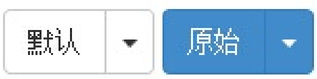
分割的按钮下拉菜单使用与下拉菜单按钮大致相同的样式，但是对下拉菜单添加了原始的功能。分割按钮的左边是原始的功能，右边是显示下拉菜单的切换。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 分割的按钮下拉菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group">
   <button type="button" class="btn btn-default">默认</button>
   <button type="button" class="btn btn-default dropdown-toggle"
      data-toggle="dropdown">
      <span class="caret"></span>
      <span class="sr-only">切换下拉菜单</span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
   </ul>
</div>
<div class="btn-group">
   <button type="button" class="btn btn-primary">原始</button>
   <button type="button" class="btn btn-primary dropdown-toggle"
      data-toggle="dropdown">
      <span class="caret"></span>
      <span class="sr-only">切换下拉菜单</span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
   </ul>
</div>

</body>
</html>
```

结果如下所示：



按钮下拉菜单的大小

您可以使用带有各种大小按钮的下拉菜单：**.btn-large**、**.btn-sm** 或 **.btn-xs**。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮下拉菜单的大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group">
   <button type="button" class="btn btn-default dropdown-toggle btn
      data-toggle="dropdown">
      默认 <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
   </ul>
</div>
<div class="btn-group">
   <button type="button" class="btn btn-primary dropdown-toggle btn
      data-toggle="dropdown">
      原始 <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
   </ul>
</div>
<div class="btn-group">
   <button type="button" class="btn btn-success dropdown-toggle btn
      data-toggle="dropdown">
      成功 <span class="caret"></span>
   </button>
   <ul class="dropdown-menu" role="menu">
      <li><a href="#">功能</a></li>
      <li><a href="#">另一个功能</a></li>
      <li><a href="#">其他</a></li>
      <li class="divider"></li>
      <li><a href="#">分离的链接</a></li>
```

```
    </ul>
</div>

</body>
</html>
```

结果如下所示：



按钮上拉菜单

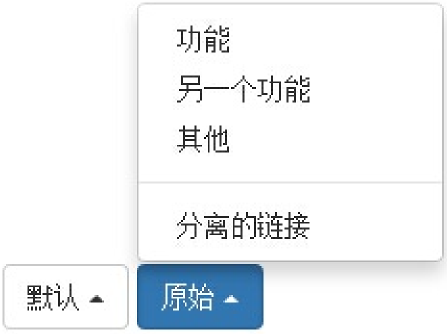
菜单也可以往上拉伸的，只需要简单地向父 **.btn-group** 容器添加 **.dropup** 即可。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮上拉菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="row" style="margin-left:50px; margin-top:200px">
   <div class="btn-group dropup">
      <button type="button" class="btn btn-default dropdown-toggle'
         data-toggle="dropdown">
         默认 <span class="caret"></span>
      </button>
      <ul class="dropdown-menu" role="menu">
         <li><a href="#">功能</a></li>
         <li><a href="#">另一个功能</a></li>
         <li><a href="#">其他</a></li>
         <li class="divider"></li>
         <li><a href="#">分离的链接</a></li>
      </ul>
   </div>
   <div class="btn-group dropup">
      <button type="button" class="btn btn-primary dropdown-toggle'
         data-toggle="dropdown">
         原始 <span class="caret"></span>
      </button>
      <ul class="dropdown-menu" role="menu">
         <li><a href="#">功能</a></li>
         <li><a href="#">另一个功能</a></li>
         <li><a href="#">其他</a></li>
         <li class="divider"></li>
         <li><a href="#">分离的链接</a></li>
      </ul>
   </div>
</div>

</body>
</html>
```

结果如下所示：


功能
另一个功能
其他
分离的链接
默认
原始

Bootstrap 输入框组

本章将讲解 Bootstrap 支持的另一个特性，输入框组。输入框组扩展自 表单控件。使用输入框组，您可以很容易地向基于文本的输入框添加作为前缀和后缀的文本或按钮。

通过向输入域添加前缀和后缀的内容，您可以向用户输入添加公共的元素。例如，您可以添加美元符号，或者在 Twitter 用户名前添加 @，或者应用程序接口所需要的其他公共的元素。

向 **.form-control** 添加前缀或后缀元素的步骤如下：

- 把前缀后后缀元素放在一个带有 class **.input-group** 的 <div> 中。
- 接着，在相同的 <div> 内，在 class 为 **.input-group-addon** 的 <span> 内放置额外的内容。
- 把该 <span> 放置在 <input> 元素的前面或者后面。

> 为了保持跨浏览器的兼容性，请避免使用 <select> 元素，因为它们在 WebKit 浏览器中不能完全渲染出效果。也不要直接向表单组应用输入框组的 class，输入框组是一个孤立的组件。

基本的输入框组

下面的实例演示了基本的输入框组：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本的输入框组</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div style="padding: 100px 100px 10px;">
   <form class="bs-example bs-example-form" role="form">
      <div class="input-group">
         <span class="input-group-addon">@</span>
         <input type="text" class="form-control" placeholder="twitt
      </div>
      <br>

      <div class="input-group">
         <input type="text" class="form-control">
         <span class="input-group-addon">.00</span>
      </div>
      <br>
      <div class="input-group">
         <span class="input-group-addon">$</span>
         <input type="text" class="form-control">
         <span class="input-group-addon">.00</span>
      </div>
   </form>
</div>

</body>
</html>
```

结果如下所示：



输入框组的大小

您可以通过向 **.input-group** 添加相对表单大小的 class（比如 **.input-group-lg**、**input-group-sm**、**input-group-xs**）来改变输入框组的大小。输入框中的内容会自动调整大小。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 输入框组的大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

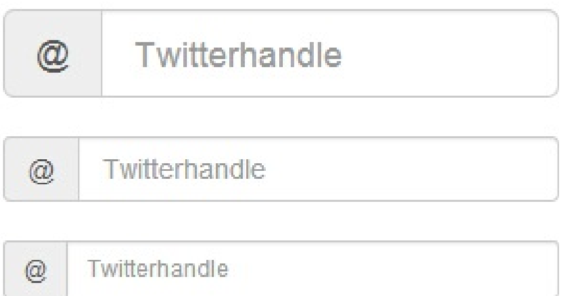
<div style="padding: 100px 100px 10px;">
   <form class="bs-example bs-example-form" role="form">
      <div class="input-group input-group-lg">
         <span class="input-group-addon">@</span>
         <input type="text" class="form-control" placeholder="Twit
      </div><br>

      <div class="input-group">
         <span class="input-group-addon">@</span>
         <input type="text" class="form-control" placeholder="Twit
      </div><br>

      <div class="input-group input-group-sm">
         <span class="input-group-addon">@</span>
         <input type="text" class="form-control" placeholder="Twit
      </div>
   </form>
</div>

</body>
</html>
```

结果如下所示：



复选框和单选插件

您可以把复选框和单选插件作为输入框组的前缀或者后缀元素，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 输入框组的复选框和单选插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div style="padding: 100px 100px 10px;">
   <form class="bs-example bs-example-form" role="form">
      <div class="row">
         <div class="col-lg-6">
            <div class="input-group">
               <span class="input-group-addon">
                  <input type="checkbox">
               </span>
               <input type="text" class="form-control">
            </div><!-- /input-group -->
         </div><!-- /.col-lg-6 --><br>
         <div class="col-lg-6">
            <div class="input-group">
               <span class="input-group-addon">
                  <input type="radio">
               </span>
               <input type="text" class="form-control">
            </div><!-- /input-group -->
         </div><!-- /.col-lg-6 -->
      </div><!-- /.row -->
   </form>
</div>

</body>
</html>
```

结果如下所示：



按钮插件

您也可以把按钮作为输入框组的前缀或者后缀元素，这个时候您就不是添加 **.input-group-addon** class，您需要使用 class **.input-group-btn** 来包裹按钮。这是必需的，因为默认的浏览器样式不会被重写。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 输入框组的按钮插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div style="padding: 100px 100px 10px;">
   <form class="bs-example bs-example-form" role="form">
      <div class="row">
         <div class="col-lg-6">
            <div class="input-group">
               <span class="input-group-btn">
                  <button class="btn btn-default" type="button">
                     Go!
                  </button>
               </span>
               <input type="text" class="form-control">
            </div><!-- /input-group -->
         </div><!-- /.col-lg-6 --><br>
         <div class="col-lg-6">
            <div class="input-group">
               <input type="text" class="form-control">
               <span class="input-group-btn">
                  <button class="btn btn-default" type="button">
                     Go!
                  </button>
               </span>
            </div><!-- /input-group -->
         </div><!-- /.col-lg-6 -->
      </div><!-- /.row -->
   </form>
</div>

</body>
</html>
```

结果如下所示：



带有下拉菜单的按钮

在输入框组中添加带有下拉菜单的按钮，只需要简单地在一个 **.input-group-btn** class 中包裹按钮和下拉菜单即可，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 输入框组的下拉菜单按钮</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div style="padding: 100px 100px 10px;">
   <form class="bs-example bs-example-form" role="form">
      <div class="row">
         <div class="col-lg-6">
            <div class="input-group">
               <div class="input-group-btn">
                  <button type="button" class="btn btn-default
                     dropdown-toggle" data-toggle="dropdown">
                     下拉菜单
                     <span class="caret"></span>
                  </button>
                  <ul class="dropdown-menu">
                     <li><a href="#">功能</a></li>
                     <li><a href="#">另一个功能</a></li>
                     <li><a href="#">其他</a></li>
                     <li class="divider"></li>
                     <li><a href="#">分离的链接</a></li>
                  </ul>
               </div><!-- /btn-group -->
               <input type="text" class="form-control">
            </div><!-- /input-group -->
         </div><!-- /.col-lg-6 --><br>
         <div class="col-lg-6">
            <div class="input-group">
               <input type="text" class="form-control">
               <div class="input-group-btn">
                  <button type="button" class="btn btn-default
                     dropdown-toggle" data-toggle="dropdown">
                     下拉菜单
```

```
                     <span class="caret"></span>
                  </button>
                  <ul class="dropdown-menu pull-right">
                     <li><a href="#">功能</a></li>
                     <li><a href="#">另一个功能</a></li>
                     <li><a href="#">其他</a></li>
                     <li class="divider"></li>
                     <li><a href="#">分离的链接</a></li>
                  </ul>
               </div><!-- /btn-group -->
            </div><!-- /input-group -->
         </div><!-- /.col-lg-6 -->
      </div><!-- /.row -->
   </form>
</div>

</body>
</html>
```

结果如下所示：



分割的下拉菜单按钮

在输入框组中添加带有下拉菜单的分割按钮，使用与下拉菜单按钮大致相同的样式，但是对下拉菜单添加了主要的功能，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 输入框组中分割的下拉菜单按钮</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div style="padding: 100px 100px 10px;">
   <form class="bs-example bs-example-form" role="form">
      <div class="row">
         <div class="col-lg-6">
            <div class="input-group">
               <div class="input-group-btn">
```

```
                    <button type="button" class="btn btn-default"
                       tabindex="-1">下拉菜单
                    </button>
                    <button type="button" class="btn btn-default
                       dropdown-toggle" data-toggle="dropdown" tabinc
                       <span class="caret"></span>
                       <span class="sr-only">切换下拉菜单</span>
                    </button>
                    <ul class="dropdown-menu">
                       <li><a href="#">功能</a></li>
                       <li><a href="#">另一个功能</a></li>
                       <li><a href="#">其他</a></li>
                       <li class="divider"></li>
                       <li><a href="#">分离的链接</a></li>
                    </ul>
                 </div><!-- /btn-group -->
                 <input type="text" class="form-control">
              </div><!-- /input-group -->
           </div><!-- /.col-lg-6 --><br>
           <div class="col-lg-6">
              <div class="input-group">
                 <input type="text" class="form-control">
                 <div class="input-group-btn">
                    <button type="button" class="btn btn-default"
                       tabindex="-1">下拉菜单
                    </button>
                    <button type="button" class="btn btn-default
                       dropdown-toggle" data-toggle="dropdown" tabinc
                       <span class="caret"></span>
                       <span class="sr-only">切换下拉菜单</span>
                    </button>
                    <ul class="dropdown-menu pull-right">
                       <li><a href="#">功能</a></li>
                       <li><a href="#">另一个功能</a></li>
                       <li><a href="#">其他</a></li>
                       <li class="divider"></li>
                       <li><a href="#">分离的链接</a></li>
                    </ul>
                 </div><!-- /btn-group -->
              </div><!-- /input-group -->
           </div><!-- /.col-lg-6 -->
        </div><!-- /.row -->
     </form>
  </div>

  </body>
  </html>
```

结果如下所示：


下拉菜单
下拉菜单

Bootstrap 导航元素

本章我们将讲解 Bootstrap 提供的用于定义导航元素的一些选项。它们使用相同的标记和基类 **.nav**。Bootstrap 也提供了一个用于共享标记和状态的帮助器类。改变修饰的 class，可以在不同的样式间进行切换。

表格导航或标签

创建一个标签式的导航菜单：

- 以一个带有 class **.nav** 的无序列表开始。
- 添加 class **.nav-tabs**。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标签式的导航菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>标签式的导航菜单</p>
<ul class="nav nav-tabs">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



胶囊式的导航菜单

基本的胶囊式导航菜单

如果需要把标签改成胶囊的样式，只需要使用 class **.nav-pills** 代替 **.nav-tabs** 即可，其他的步骤与上面相同。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本的胶囊式导航菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>基本的胶囊式导航菜单</p>
<ul class="nav nav-pills">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



垂直的胶囊式导航菜单

您可以在使用 class **.nav**、**.nav-pills** 的同时使用 class **.nav-stacked**，让胶囊垂直堆叠。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 垂直的胶囊式导航菜单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>垂直的胶囊式导航菜单</p>
<ul class="nav nav-pills nav-stacked">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：

垂直的胶囊式导航菜单

Home

SVN

iOS

VB.Net

Java

PHP

两端对齐的导航

您可以在屏幕宽度大于 768px 时，通过在分别使用 **.nav**、**.nav-tabs** 或 **.nav**、**.nav-pills** 的同时使用 class **.nav-justified**，让标签式或胶囊式导航菜单与父元素等宽。在更小的屏幕上，导航链接会堆叠。

下面的实例演示了这点：

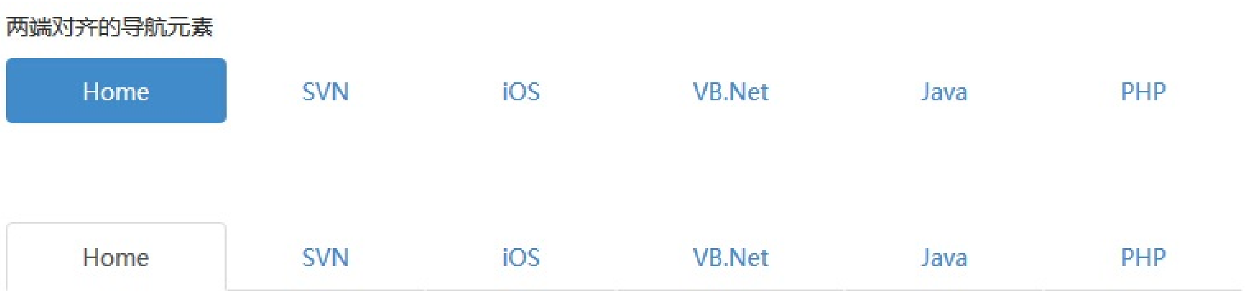
```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 两端对齐的导航元素</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>两端对齐的导航元素</p>
<ul class="nav nav-pills nav-justified">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul><br><br><br>

<ul class="nav nav-tabs nav-justified">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



禁用链接

对每个 **.nav** class，如果添加了 **.disabled** class，则会创建一个灰色的链接，同时禁用了该链接的 **:hover** 状态，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 导航元素中的禁用链接</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>导航元素中的禁用链接</p>
<ul class="nav nav-pills">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li class="disabled"><a href="#">iOS（禁用链接）</a></li>
   <li><a href="#">VB.Net</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul><br><br>

<ul class="nav nav-tabs">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li  class="disabled"><a href="#">VB.Net（禁用链接）</a></li>
   <li><a href="#">Java</a></li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



> 该 class 只会改变 <a> 的外观，不会改变它的功能。在这里，您需要使用自定义的 JavaScript 来禁用链接。

下拉菜单

导航菜单与下拉菜单使用相似的语法。默认情况下，列表项的锚与一些数据属性协同合作来触发带有 **.dropdown-menu** class 的无序列表。

带有下拉菜单的标签

向标签添加下拉菜单的步骤如下：

- 以一个带有 class **.nav** 的无序列表开始。
- 添加 class **.nav-tabs**。
- 添加带有 **.dropdown-menu** class 的无序列表。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 带有下拉菜单的标签</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>带有下拉菜单的标签</p>
<ul class="nav nav-tabs">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li class="dropdown">
      <a class="dropdown-toggle" data-toggle="dropdown" href="#">
         Java <span class="caret"></span>
      </a>
      <ul class="dropdown-menu">
         <li><a href="#">Swing</a></li>
         <li><a href="#">jMeter</a></li>
         <li><a href="#">EJB</a></li>
         <li class="divider"></li>
         <li><a href="#">分离的链接</a></li>
      </ul>
   </li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



带有下拉菜单的胶囊

步骤与创建带有下拉菜单的标签相同，只是需要把 **.nav-tabs** class 改为 **.nav-pills**，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 带有下拉菜单的胶囊</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>带有下拉菜单的胶囊</p>
<ul class="nav nav-pills">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li class="dropdown">
      <a class="dropdown-toggle" data-toggle="dropdown" href="#">
         Java <span class="caret"></span>
      </a>
      <ul class="dropdown-menu">
         <li><a href="#">Swing</a></li>
         <li><a href="#">jMeter</a></li>
         <li><a href="#">EJB</a></li>
         <li class="divider"></li>
         <li><a href="#">分离的链接</a></li>
      </ul>
   </li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



Bootstrap 导航栏

导航栏是一个很好的功能，是 Bootstrap 网站的一个突出特点。导航栏是响应式元组件就，作为应用程序或网站的导航标题。导航栏在移动设备的视图中是折叠的，随着可用视口宽度的增加，导航栏也会水平展开。在 Bootstrap 导航栏的核心中，导航栏包括了为站点名称和基本的导航定义样式。

默认的导航栏

创建一个默认的导航栏的步骤如下：

- 向 <nav> 标签添加 class **.navbar**、**.navbar-default**。
- 向上面的元素添加 **role="navigation"**，有助于增加可访问性。
- 向 <div> 元素添加一个标题 class **.navbar-header**，内部包含了带有 class **navbar-brand** 的 <a> 元素。这会让文本看起来更大一号。
- 为了向导航栏添加链接，只需要简单地添加带有 class **.nav**、**.navbar-nav** 的无序列表即可。

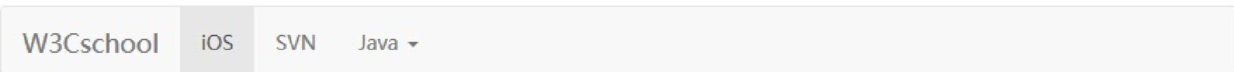
下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的导航栏</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java
               <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



响应式的导航栏

为了给导航栏添加响应式特性，您要折叠的内容必须包裹在带有 classes **.collapse**、**.navbar-collapse** 的 <div> 中。折叠起来的导航栏实际上是一个带有 class **.navbar-toggle** 及两个 data- 元素的按钮。第一个是 **data-toggle**，用于告诉 JavaScript 需要对按钮做什么，第二个是 **data-target**，指示要切换到哪一个元素。三个带有 class **.icon-bar** 的 <span> 创建所谓的汉堡按钮。这些会切换为 **.nav-collapse** <div> 中的元素。为了实现以上这些功能，您必须包含 Bootstrap 折叠（Collapse）插件。

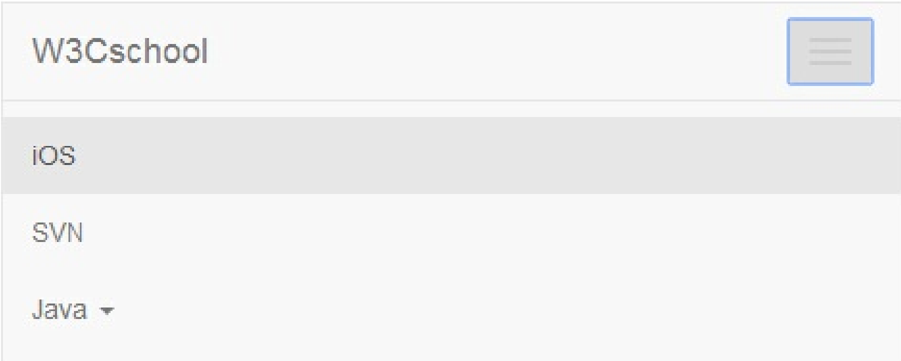
下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 响应式的导航栏</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="col
         data-target="#example-navbar-collapse">
         <span class="sr-only">切换导航</span>
         <span class="icon-bar"></span>
         <span class="icon-bar"></span>
         <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div class="collapse navbar-collapse" id="example-navbar-collaps
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



导航栏中的表单

导航栏中的表单不是使用 Bootstrap 表单 章节中所讲到的默认的 class，它是使用 **.navbar-form** class。这确保了表单适当的垂直对齐和在较窄的视口中折叠的行为。使用对齐方式选项（这将在组件对齐方式部分进行详细讲解）来决定导航栏中的内容放置在哪里。

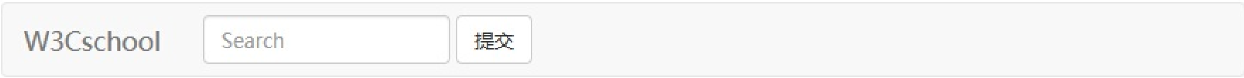
下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 导航栏中的表单</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <form class="navbar-form navbar-left" role="search">
         <div class="form-group">
            <input type="text" class="form-control" placeholder="Se
         </div>
         <button type="submit" class="btn btn-default">提交</button
      </form>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



导航栏中的按钮

您可以使用 class **.navbar-btn** 向不在 <form> 中的 <button> 元素添加按钮，按钮在导航栏上垂直居中。**.navbar-btn** 可被使用在 <a> 和 <input> 元素上。

> 不要在 **.navbar-nav** 内的 <a> 元素上使用 **.navbar-btn**，因为它不是标准的 button class。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 导航栏中的按钮</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <form class="navbar-form navbar-left" role="search">
         <div class="form-group">
            <input type="text" class="form-control" placeholder="Se
         </div>
         <button type="submit" class="btn btn-default">提交按钮</but
      </form>
      <button type="button" class="btn btn-default navbar-btn">
         导航栏按钮
      </button>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



导航栏中的文本

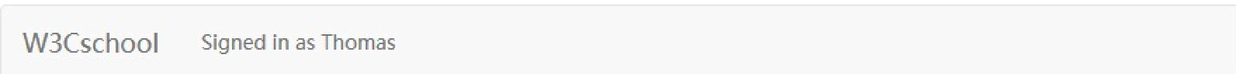
如果需要在导航中包含文本字符串，请使用 class **.navbar-text**。这通常与 <p> 标签一起使用，确保适当的前导和颜色。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 导航栏中的文本</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <p class="navbar-text">Signed in as Thomas</p>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



非导航链接

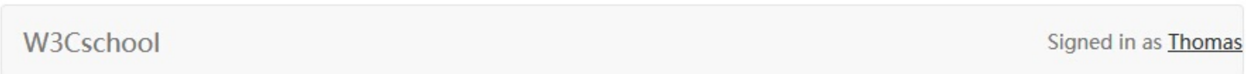
如果您不想在常规的导航栏导航组件内使用标准的链接，那么请使用 class **navbar-link** 来为默认的和倒转的导航栏选项添加适当的颜色，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 非导航链接</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <p class="navbar-text navbar-right">Signed in as
         <a href="#" class="navbar-link">Thomas</a>
      </p>
   </div>
</div>

</body>
</html>
```

结果如下所示：



组件对齐方式

您可以使用实用工具 class **.navbar-left** 或 **.navbar-right** 向左或向右对齐导航栏中的导航链接、表单、按钮或文本这些组件。这两个 class 都会在指定的方向上添加 CSS 浮动。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 组件对齐方式</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

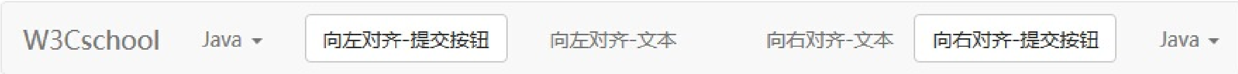
<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
```

```
    </div>
    <div>
       <!--向左对齐-->
       <ul class="nav navbar-nav navbar-left">
          <li class="dropdown">
             <a href="#" class="dropdown-toggle" data-toggle="dropdo
                Java
                <b class="caret"></b>
             </a>
             <ul class="dropdown-menu">
                <li><a href="#">jmeter</a></li>
                <li><a href="#">EJB</a></li>
                <li><a href="#">Jasper Report</a></li>
                <li class="divider"></li>
                <li><a href="#">分离的链接</a></li>
                <li class="divider"></li>
                <li><a href="#">另一个分离的链接</a></li>
             </ul>
          </li>
       </ul>
       <form class="navbar-form navbar-left" role="search">
          <button type="submit" class="btn btn-default">
             向左对齐-提交按钮
          </button>
       </form>
       <p class="navbar-text navbar-left">向左对齐-文本</p>
       <!--向右对齐-->
       <ul class="nav navbar-nav navbar-right">
          <li class="dropdown">
             <a href="#" class="dropdown-toggle" data-toggle="dropdo
                Java <b class="caret"></b>
             </a>
             <ul class="dropdown-menu">
                <li><a href="#">jmeter</a></li>
                <li><a href="#">EJB</a></li>
                <li><a href="#">Jasper Report</a></li>
                <li class="divider"></li>
                <li><a href="#">分离的链接</a></li>
                <li class="divider"></li>
                <li><a href="#">另一个分离的链接</a></li>
             </ul>
          </li>
       </ul>
       <form class="navbar-form navbar-right" role="search">
          <button type="submit" class="btn btn-default">
             向右对齐-提交按钮
          </button>
       </form>
       <p class="navbar-text navbar-right">向右对齐-文本</p>
    </div>
 </nav>

 </body>
```

```
</html>
```

结果如下所示：



固定到顶部

Bootstrap 导航栏可以动态定位。默认情况下，它是块级元素，它是基于在 HTML 中放置的位置定位的。通过一些帮助器类，您可以把它放置在页面的顶部或者底部，或者您可以让它成为随着页面一起滚动的静态导航栏。

如果您想要让导航栏固定在页面的顶部，请向 **.navbar class** 添加 class **.navbar-fixed-top**。下面的实例演示了这点：

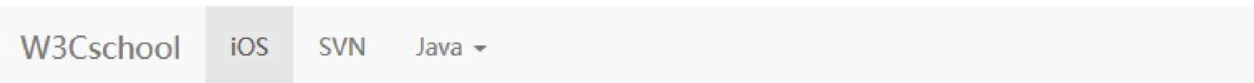
> 为了防止导航栏与页面主体中的其他内容的顶部相交错，请向 <body> 标签添加至少 50 像素的内边距（padding），内边距的值可以根据您的需要进行设置。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 固定到顶部</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default navbar-fixed-top" role="navigati
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



固定到底部

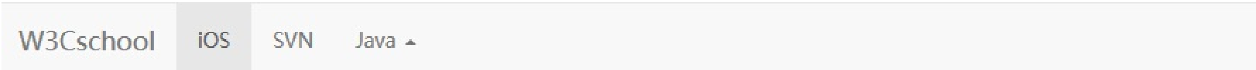
如果您想要让导航栏固定在页面的底部，请向 **.navbar class** 添加 class **.navbar-fixed-bottom**。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 固定到底部</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default navbar-fixed-bottom" role="naviga
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



静态的顶部

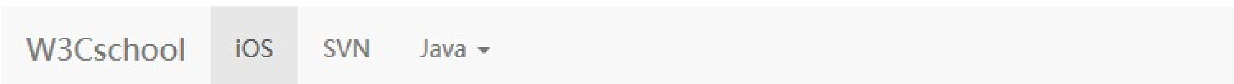
如需创建能随着页面一起滚动的导航栏，请添加 **.navbar-static-top** class。该 class 不要求向 <body> 添加内边距（padding）。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 静态的顶部</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default navbar-static-top" role="navigati
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



倒置的导航栏

为了创建一个带有黑色背景白色文本的倒置的导航栏，只需要简单地向 **.navbar** class 添加 **.navbar-inverse** class 即可，如下面的实例所示：

> 为了防止导航栏与页面主体中的其他内容的顶部相交错，请向 <body> 标签添加至少 50 像素的内边距（padding），内边距的值可以根据您的需要进行设置。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 倒置的导航栏</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-inverse" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



Bootstrap 面包屑导航（Breadcrumbs）

面包屑导航（Breadcrumbs）是一种基于网站层次信息的显示方式。以博客为例，面包屑导航可以显示发布日期、类别或标签。它们表示当前页面在导航层次结构内的位置。

Bootstrap 中的面包屑导航（Breadcrumbs）是一个简单的带有 **.breadcrumb** class 的无序列表。分隔符会通过 CSS（bootstrap.min.css）中下面所示的 class 自动被添加：

```
.breadcrumb > li + li:before {
    color: #CCCCCC;
    content: "/ ";
    padding: 0 5px;
}
```

下面的实例演示了面包屑导航：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 面包屑导航</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ol class="breadcrumb">
  <li><a href="#">Home</a></li>
  <li><a href="#">2013</a></li>
  <li class="active">十一月</li>
</ol>

</body>
</html>
```

结果如下所示：

Home / 2013 / 十一月

Bootstrap 分页

本章将讲解 Bootstrap 支持的分页特性。分页（Pagination），是一种无序列表，Bootstrap 像处理其他界面元素一样处理分页。

分页（Pagination）

下表列出了 Bootstrap 提供的处理分页的 class。

.pagination

添加该 class 来在页面上显示分页。

```
&lt;ul class="pagination"&gt;
  &lt;li&gt;&lt;a href="#"&gt;&laquo;&lt;/a&gt;&lt;/li&gt;
  &lt;li&gt;&lt;a href="#"&gt;1&lt;/a&gt;&lt;/li&gt;
  .......
&lt;/ul&gt;
```

.disabled, .active

您可以自定义链接，通过使用 **.disabled** 来定义不可点击的链接，通过使用 **.active** 来指示当前的页面。

```
&lt;ul class="pagination"&gt;
  &lt;li class="disabled"&gt;&lt;a href="#"&gt;&laquo;&lt;/a&gt;&lt
  &lt;li class="active"&gt;&lt;a href="#"&gt;1&lt;span class="sr-or
  .......
&lt;/ul&gt;
```

.pagination-lg, .pagination-sm

使用这些 class 来获取不同大小的项。

```
&lt;ul class="pagination pagination-lg"&gt;...&lt;/ul&gt;
&lt;ul class="pagination"&gt;...&lt;/ul&gt;
&lt;ul class="pagination pagination-sm"&gt;...&lt;/ul&gt;
```

默认的分页

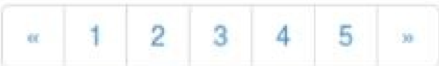
下面的实例演示了上表中所讨论的 class **.pagination** 的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的分页</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="pagination">
  <li><a href="#">&laquo;</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li><a href="#">&raquo;</a></li>
</ul>

</body>
</html>
```

结果如下所示：



分页的状态

下面的实例演示了上表中所讨论的 class **.disabled**、**.active** 的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 分页的状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="pagination">
  <li><a href="#">&laquo;</a></li>
  <li class="active"><a href="#">1</a></li>
  <li class="disabled"><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li><a href="#">&raquo;</a></li>
</ul>

</body>
</html>
```

结果如下所示：



分页的大小

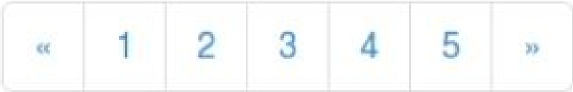
下面的实例演示了上表中所讨论的 class **.pagination-*** 的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 分页的大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="pagination pagination-lg">
  <li><a href="#">&laquo;</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li><a href="#">&raquo;</a></li>
</ul><br>
<ul class="pagination">
  <li><a href="#">&laquo;</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li><a href="#">&raquo;</a></li>
</ul><br>
<ul class="pagination pagination-sm">
 <li><a href="#">&laquo;</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li><a href="#">&raquo;</a></li>
</ul>

</body>
</html>
```

结果如下所示：







翻页（Pager）

如果您想要创建一个简单的分页链接为用户提供导航，可通过翻页来实现。与分页链接一样，翻页也是无序列表。默认情况下，链接是居中显示。下表列出了 Bootstrap 处理翻页的 class。

.pager

添加该 class 来获得翻页链接。

```
&lt;ul class="pager"&gt;
  &lt;li&gt;&lt;a href="#"&gt;Previous&lt;/a&gt;&lt;/li&gt;
  &lt;li&gt;&lt;a href="#"&gt;Next&lt;/a&gt;&lt;/li&gt;
&lt;/ul&gt;
```

.previous, .next

使用 class **.previous** 把链接向左对齐，使用 **.next** 把链接向右对齐。

```
&lt;ul class="pager"&gt;
  &lt;li class="previous"&gt;&lt;a href="#"&gt;&larr; Older&lt;/a&g
  &lt;li class="next"&gt;&lt;a href="#"&gt;Newer &rarr;&lt;/a&gt;&l
&lt;/ul&gt;
```

.disabled

添加该 class 来获得一个颜色变淡的外观。

```
&lt;ul class="pager"&gt;
  &lt;li class="previous disabled"&gt;&lt;a href="#"&gt;&larr; Olde
  &lt;li class="next"&gt;&lt;a href="#"&gt;Newer &rarr;&lt;/a&gt;&l
&lt;/ul&gt;
```

默认的翻页

下面的实例演示了上表中所讨论的 class **.pager** 的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的翻页</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="pager">
  <li><a href="#">Previous</a></li>
  <li><a href="#">Next</a></li>
</ul>

</body>
</html>
```

结果如下所示：



对齐的链接

下面的实例演示了上表中所讨论的 class **.previous**、**.next** 的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 翻页中对齐的链接</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="pager">
  <li class="previous"><a href="#">&larr; Older</a></li>
  <li class="next"><a href="#">Newer &rarr;</a></li>
</ul>

</body>
</html>
```

结果如下所示：



翻页的状态

下面的实例演示了上表中所讨论的 class **.disabled** 的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 翻页的状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="pager">
  <li class="previous disabled"><a href="#">&larr; Older</a></li>
  <li class="next"><a href="#">Newer &rarr;</a></li>
</ul>

</body>
</html>
```

结果如下所示：



Bootstrap 标签

本章将讲解 Bootstrap 标签。标签可用于计数、提示或页面上其他的标记显示。使用 class **.label** 来显示标签，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标签</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h1>Example Heading <span class="label label-default">Label</span><
<h2>Example Heading <span class="label label-default">Label</span><
<h3>Example Heading <span class="label label-default">Label</span><
<h4>Example Heading <span class="label label-default">Label</span><

</body>
</html>
```

结果如下所示：



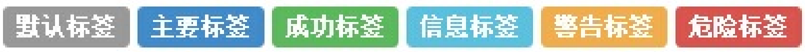
您可以使用修饰的 class **label-default**、**label-primary**、**label-success**、**label-info**、**label-warning**、**label-danger** 来改变标签的外观，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标签的变体</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<span class="label label-default">默认标签</span>
<span class="label label-primary">主要标签</span>
<span class="label label-success">成功标签</span>
<span class="label label-info">信息标签</span>
<span class="label label-warning">警告标签</span>
<span class="label label-danger">危险标签</span>

</body>
</html>
```

结果如下所示：



Bootstrap 徽章（Badges）

本章将讲解 Bootstrap 徽章（Badges）。徽章与标签相似，主要的区别在于徽章的边角更加圆滑。

徽章（Badges）主要用于突出显示新的或未读的项。如需使用徽章，只需要把 **<span class="badge">** 添加到链接、Bootstrap 导航等这些元素上即可。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 徽章（Badges）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<a href="#">Mailbox <span class="badge">50</span></a>

</body>
</html>
```

结果如下所示：



当没有新的或未读的项时，通过 CSS 的 **:empty** 选择器，徽章会折叠起来，表示里边没有内容。

激活导航状态

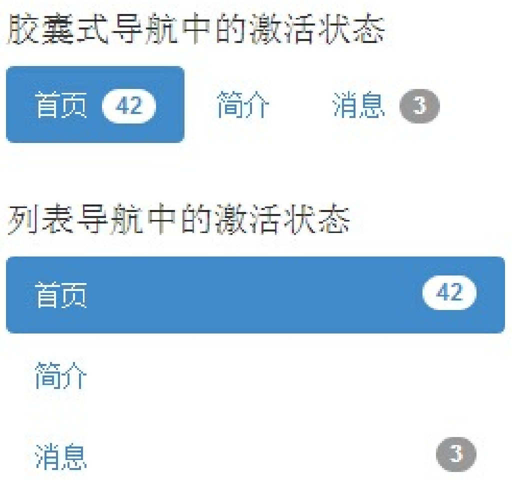
您可以在激活状态的胶囊式导航和列表导航中放置徽章。通过使用 **<span class="badge">** 来激活链接，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 激活导航状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h4>胶囊式导航中的激活状态</h4>
<ul class="nav nav-pills">
   <li class="active"><a href="#">首页 <span class="badge">42</span
   <li><a href="#">简介</a></li>
   <li><a href="#">消息 <span class="badge">3</span></a></li>
</ul>
<br>
<h4>列表导航中的激活状态</h4>
<ul class="nav nav-pills nav-stacked" style="max-width: 260px;">
   <li class="active">
      <a href="#">
         <span class="badge pull-right">42</span>
         首页
      </a>
   </li>
   <li><a href="#">简介</a></li>
   <li>
      <a href="#">
         <span class="badge pull-right">3</span>
         消息
      </a>
   </li>
</ul>

</body>
</html>
```

结果如下所示：


胶囊式导航中的激活状态
首页 42
简介
消息 3
列表导航中的激活状态
首页 42
简介
消息 3

Bootstrap 超大屏幕（Jumbotron）

本章将讲解 Bootstrap 支持的另一个特性，超大屏幕（Jumbotron）。顾名思义该组件可以增加标题的大小，并为登陆页面内容添加更多的外边距（margin）。使用超大屏幕（Jumbotron）的步骤如下：

- 创建一个带有 class **.jumbotron**. 的容器 <div>。
- 除了更大的 <h1>，字体粗细 *font-weight* 被减为 200px。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 超大屏幕（Jumbotron）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
   <div class="jumbotron">
      <h1>欢迎登陆页面！</h1>
      <p>这是一个超大屏幕（Jumbotron）的实例。</p>
      <p><a class="btn btn-primary btn-lg" role="button">
         学习更多</a>
      </p>
   </div>
</div>

</body>
</html>
```

结果如下所示：



为了获得占用全部宽度且不带圆角的超大屏幕，请在所有的 **.container** class 外使用 **.jumbotron** class，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 超大屏幕（Jumbotron）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="jumbotron">
   <div class="container">
      <h1>欢迎登陆页面！</h1>
      <p>这是一个超大屏幕（Jumbotron）的实例。</p>
      <p><a class="btn btn-primary btn-lg" role="button">
         学习更多</a>
      </p>
   </div>
</div>

</body>
</html>
```

结果如下所示：


欢迎登陆页面！
这是一个超大屏幕（Jumbotron）的实例。
学习更多

Bootstrap 页面标题（Page Header）

页面标题（Page Header）是个不错的功能，它会在网页标题四周添加适当的间距。当一个网页中有多个标题且每个标题之间需要添加一定的间距时，页面标题这个功能就显得特别有用。如需使用页面标题（Page Header），请把您的标题放置在带有 class **.page-header** 的 <div> 中：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 页面标题</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="page-header">
   <h1>页面标题实例
      <small>子标题</small>
   </h1>
</div>
<p>这是一个示例文本。这是一个示例文本。这是一个示例文本。这是一个示例文本。这是

</body>
</html>
```

结果如下所示：



Bootstrap 缩略图

本章将讲解 Bootstrap 缩略图。大多数站点都需要在网格中布局图像、视频、文本等。Bootstrap 通过缩略图为此提供了一种简便的方式。使用 Bootstrap 创建缩略图的步骤如下：

- 在图像周围添加带有 class **.thumbnail** 的 <a> 标签。
- 这会添加四个像素的内边距（padding）和一个灰色的边框。
- 当鼠标悬停在图像上时，会动画显示出图像的轮廓。

下面的实例演示了默认的缩略图：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 缩略图</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="row">
   <div class="col-sm-6 col-md-3">
      <a href="#" class="thumbnail">
         <img src="img/kittens.jpg"
         alt="通用的占位符缩略图">
      </a>
   </div>
   <div class="col-sm-6 col-md-3">
      <a href="#" class="thumbnail">
         <img src="img/kittens.jpg"
         alt="通用的占位符缩略图">
      </a>
   </div>
   <div class="col-sm-6 col-md-3">
      <a href="#" class="thumbnail">
         <img src="img/kittens.jpg"
         alt="通用的占位符缩略图">
      </a>
   </div>
   <div class="col-sm-6 col-md-3">
      <a href="#" class="thumbnail">
         <img src="img/kittens.jpg"
         alt="通用的占位符缩略图">
      </a>
   </div>
</div>

</body>
</html>
```

结果如下所示：

添加自定义的内容

现在我们有了一个基本的缩略图，我们可以向缩略图添加各种 HTML 内容，比如标题、段落或按钮。具体步骤如下：

- 把带有 class **.thumbnail** 的 <a> 标签改为 <div>。
- 在该 <div> 内，您可以添加任何您想要添加的东西。由于这是一个 <div>，我们可以使用默认的基于 span 的命名规则来调整大小。
- 如果您想要给多个图像进行分组，请把它们放置在一个无序列表中，且每个列表项向左浮动。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 自定义缩略图</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="row">
   <div class="col-sm-6 col-md-3">
      <div class="thumbnail">
         <img src="img/kittens.jpg"
         alt="通用的占位符缩略图">
      </div>
      <div class="caption">
         <h3>缩略图标签</h3>
         <p>一些示例文本。一些示例文本。</p>
         <p>
            <a href="#" class="btn btn-primary" role="button">
               按钮
            </a>
            <a href="#" class="btn btn-default" role="button">
               按钮
            </a>
         </p>
      </div>
   </div>
   <div class="col-sm-6 col-md-3">
      <div class="thumbnail">
         <img src="img/kittens.jpg"
         alt="通用的占位符缩略图">
      </div>
      <div class="caption">
         <h3>缩略图标签</h3>
         <p>一些示例文本。一些示例文本。</p>
         <p>
```

```
                <a href="#" class="btn btn-primary" role="button">
                   按钮
                </a>
                <a href="#" class="btn btn-default" role="button">
                   按钮
                </a>
             </p>
          </div>
       </div>
       <div class="col-sm-6 col-md-3">
          <div class="thumbnail">
             <img src="img/kittens.jpg"
             alt="通用的占位符缩略图">
          </div>
          <div class="caption">
             <h3>缩略图标签</h3>
             <p>一些示例文本。一些示例文本。</p>
             <p>
                <a href="#" class="btn btn-primary" role="button">
                   按钮
                </a>
                <a href="#" class="btn btn-default" role="button">
                   按钮
                </a>
             </p>
          </div>
       </div>
       <div class="col-sm-6 col-md-3">
          <div class="thumbnail">
             <img src="img/kittens.jpg"
             alt="通用的占位符缩略图">
          </div>
          <div class="caption">
             <h3>缩略图标签</h3>
             <p>一些示例文本。一些示例文本。</p>
             <p>
                <a href="#" class="btn btn-primary" role="button">
                   按钮
                </a>
                <a href="#" class="btn btn-default" role="button">
                   按钮
                </a>
             </p>
          </div>
       </div>
    </div>

    </body>
    </html>
```

结果如下所示：


缩略图标签
一些示例文本。一些示例文本。
按钮
按钮
缩略图标签
一些示例文本。一些示例文本。
按钮
按钮
缩略图标签
一些示例文本。一些示例文本。
按钮
按钮
缩略图标签
一些示例文本。一些示例文本。
按钮
按钮

Bootstrap 警告（Alerts）

本章将讲解警告（Alerts）以及 Bootstrap 所提供的用于警告的 class。警告（Alerts）向用户提供了一种定义消息样式的方式。它们为典型的用户操作提供了上下文信息反馈。

您可以为警告框添加一个可选的关闭按钮。为了创建一个内联的可取消的警告框，请使用 警告（Alerts） jQuery 插件。

您可以通过创建一个 <div>，并向其添加一个 **.alert** class 和四个上下文 class（即 **.alert-success**、**.alert-info**、**.alert-warning**、**.alert-danger**）之一，来添加一个基本的警告框。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 警告（Alerts）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="alert alert-success">成功！很好地完成了提交。</div>
<div class="alert alert-info">信息！请注意这个信息。</div>
<div class="alert alert-warning">警告！请不要提交。</div>
<div class="alert alert-danger">错误！请进行一些更改。</div>

</body>
</html>
```

结果如下所示：

成功！很好地完成了提交。

信息！请注意这个信息。

警告！请不要提交。

错误！请进行一些更改。

可取消的警告（Dismissal Alerts）

创建一个可取消的警告（Dismissal Alert）步骤如下：

- 通过创建一个 <div>，并向其添加一个 **.alert** class 和四个上下文 class（即 **.alert-success**、**.alert-info**、**.alert-warning**、**.alert-danger**）之一，来添加一个基本的警告框。
- 同时向上面的 <div> class 添加可选的 **.alert-dismissable**。
- 添加一个关闭按钮。

下面的实例演示了这点：

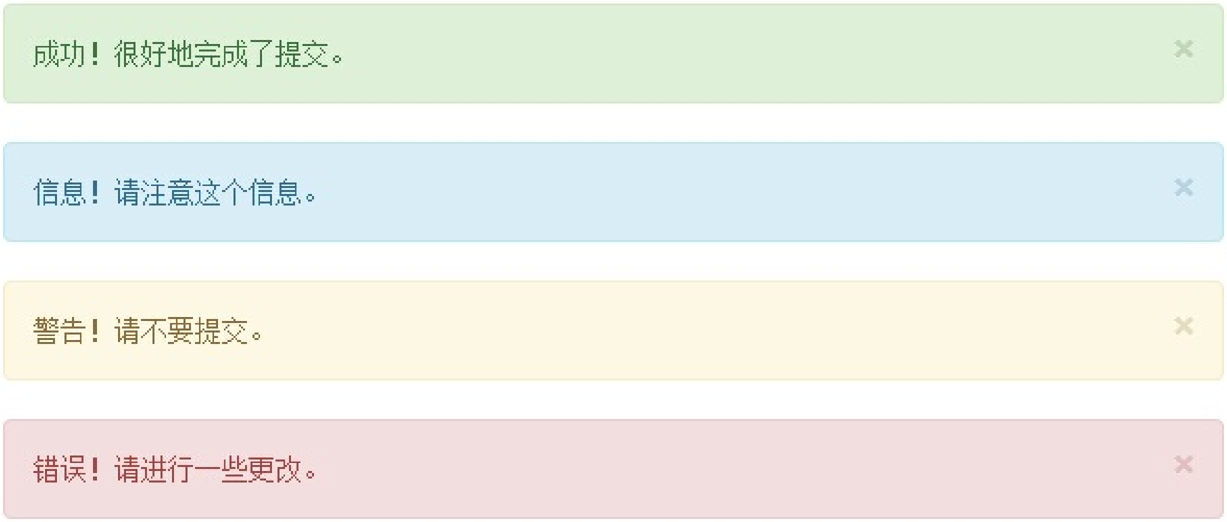
```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 可取消的警告（Dismissal Alerts）</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="alert alert-success alert-dismissable">
   <button type="button" class="close" data-dismiss="alert"
      aria-hidden="true">
      &times;
   </button>
   成功！很好地完成了提交。
</div>
<div class="alert alert-info alert-dismissable">
   <button type="button" class="close" data-dismiss="alert"
      aria-hidden="true">
      &times;
   </button>
   信息！请注意这个信息。
</div>
<div class="alert alert-warning alert-dismissable">
   <button type="button" class="close" data-dismiss="alert"
      aria-hidden="true">
      &times;
   </button>
   警告！请不要提交。
</div>
<div class="alert alert-danger alert-dismissable">
   <button type="button" class="close" data-dismiss="alert"
      aria-hidden="true">
      &times;
   </button>
   错误！请进行一些更改。
</div>

</body>
</html>
```

> 请确保使用带有 *data-dismiss="alert"* data 属性的 <button> 元素。

结果如下所示：



警告（Alerts）中的链接

在警告（Alerts）中创建链接的步骤如下：

- 通过创建一个 <div>，并向其添加一个 **.alert** class 和四个上下文 class（即 **.alert-success**、**.alert-info**、**.alert-warning**、**.alert-danger**）之一，来添加一个基本的警告框。
- 使用 **.alert-link** 实体类来快速提供带有匹配颜色的链接。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 警告（Alerts）中的链接</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="alert alert-success">
   <a href="#" class="alert-link">成功！很好地完成了提交。</a>
</div>
<div class="alert alert-info">
   <a href="#" class="alert-link">信息！请注意这个信息。</a>
</div>
<div class="alert alert-warning">
   <a href="#" class="alert-link">警告！请不要提交。</a>
</div>
<div class="alert alert-danger">
   <a href="#" class="alert-link">错误！请进行一些更改。</a>
</div>

</body>
</html>
```

结果如下所示：

成功！很好地完成了提交。

信息！请注意这个信息。

警告！请不要提交。

错误！请进行一些更改。

Bootstrap 进度条

本章将讲解 Bootstrap 进度条。在本教程中，您将看到如何使用 Bootstrap 创建加载、重定向或动作状态的进度条。

> Bootstrap 进度条使用 CSS3 过渡和动画来获得该效果。Internet Explorer 9 及之前的版本和旧版的 Firefox 不支持该特性，Opera 12 不支持动画。

默认的进度条

创建一个基本的进度条的步骤如下：

- 添加一个带有 class **.progress** 的 <div>。
- 接着，在上面的 <div> 内，添加一个带有 class **.progress-bar** 的空的 <div>。
- 添加一个带有百分比表示的宽度的 style 属性，例如 style="60%"; 表示进度条在 60% 的位置。

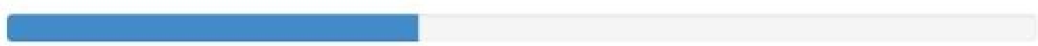
让我们看看下面的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 进度条</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="progress">
   <div class="progress-bar" role="progressbar" aria-valuenow="60"
      aria-valuemin="0" aria-valuemax="100" style="width: 40%;">
      <span class="sr-only">40% 完成</span>
   </div>
</div>

</body>
</html>
```

结果如下所示：



交替的进度条

创建不同样式的进度条的步骤如下：

- 添加一个带有 class **.progress** 的 <div>。
- 接着，在上面的 <div> 内，添加一个带有 class **.progress-bar** 和 class **progress-bar-*** 的空的 <div>。其中， 可以是 *success*、*info*、*warning*、*danger*。
- 添加一个带有百分比表示的宽度的 style 属性，例如 style="60%"; 表示进度条在 60% 的位置。

让我们看看下面的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 交替的进度条</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="progress">
   <div class="progress-bar progress-bar-success" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 90%;">
      <span class="sr-only">90% 完成（成功）</span>
   </div>
</div>
<div class="progress">
   <div class="progress-bar progress-bar-info" role="progressbar"
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 30%;">
      <span class="sr-only">30% 完成（信息）</span>
   </div>
</div>
<div class="progress">
   <div class="progress-bar progress-bar-warning" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 20%;">
      <span class="sr-only">20% 完成（警告）</span>
   </div>
</div>
<div class="progress">
   <div class="progress-bar progress-bar-danger" role="progressbar"
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 10%;">
      <span class="sr-only">10% 完成（危险）</span>
   </div>
</div>

</body>
</html>
```

结果如下所示：



条纹的进度条

创建一个条纹的进度条的步骤如下：

- 添加一个带有 class **.progress** 和 **.progress-striped** 的 <div>。
- 接着，在上面的 <div> 内，添加一个带有 class **.progress-bar** 和 class **progress-bar-*** 的空的 <div>。其中， 可以是 **success*、*info*、*warning*、*danger*。
- 添加一个带有百分比表示的宽度的 style 属性，例如 style="60%"; 表示进度条在 60% 的位置。

让我们看看下面的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 条纹的进度条</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="progress progress-striped">
   <div class="progress-bar progress-bar-success" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 90%;">
      <span class="sr-only">90% 完成（成功）</span>
   </div>
</div>
<div class="progress progress-striped">
   <div class="progress-bar progress-bar-info" role="progressbar"
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 30%;">
      <span class="sr-only">30% 完成（信息）</span>
   </div>
</div>
<div class="progress progress-striped">
   <div class="progress-bar progress-bar-warning" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 20%;">
      <span class="sr-only">20% 完成（警告）</span>
   </div>
</div>
<div class="progress progress-striped">
   <div class="progress-bar progress-bar-danger" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 10%;">
      <span class="sr-only">10% 完成（危险）</span>
   </div>
</div>

</body>
</html>
```

结果如下所示：



动画的进度条

创建一个动画的进度条的步骤如下：

- 添加一个带有 class **.progress** 和 **.progress-striped** 的 <div>。同时添加 class **.active**。
- 接着，在上面的 <div> 内，添加一个带有 class **.progress-bar** 的空的 <div>。
- 添加一个带有百分比表示的宽度的 style 属性，例如 style="60%"; 表示进度条在 60% 的位置。

这将会使条纹具有从右向左的运动感。

让我们看看下面的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 动画的进度条</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="progress progress-striped active">
   <div class="progress-bar progress-bar-success" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 40%;">
      <span class="sr-only">40% 完成</span>
   </div>
</div>

</body>
</html>
```

结果如下所示：



堆叠的进度条

您甚至可以堆叠多个进度条。把多个进度条放在相同的 **.progress** 中即可实现堆叠，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 堆叠的进度条</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="progress">
   <div class="progress-bar progress-bar-success" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 40%;">
      <span class="sr-only">40% 完成</span>
   </div>
   <div class="progress-bar progress-bar-info" role="progressbar"
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 30%;">
      <span class="sr-only">30% 完成（信息）</span>
   </div>
   <div class="progress-bar progress-bar-warning" role="progressbar
      aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
      style="width: 20%;">
      <span class="sr-only">20% 完成（警告）</span>
   </div>
</div>

</body>
</html>
```

结果如下所示：



Bootstrap 多媒体对象（Media Object）

本章我们将讲解 Bootstrap 中的多媒体对象（Media Object）。这些抽象的对象样式用于创建各种类型的组件（比如：博客评论），我们可以在组件中使用图文混排，图像可以左对齐或者右对齐。媒体对象可以用更少的代码来实现媒体对象与文字的混排。

媒体对象轻量标记、易于扩展的特性是通过向简单的标记应用 class 来实现的。你可以在 HTML 标签中添加以下两种形式来设置媒体对象：

- **.media**：该 class 允许将媒体对象里的多媒体（图像、视频、音频）浮动到内容区块的左边或者右边。
- **.media-list**：如果你需要一个列表，各项内容是无序列表的一部分，可以使用该 class。可用于评论列表与文章列表。

让我们来看看下面这个有关默认的媒体对象 .media 的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的媒体对象</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="media">
   <a class="pull-left" href="#">
      <img class="media-object" src="img/64.jpg"
      alt="媒体对象">
   </a>
   <div class="media-body">
      <h4 class="media-heading">媒体标题</h4>
      这是一些示例文本。这是一些示例文本。
      这是一些示例文本。这是一些示例文本。
      这是一些示例文本。这是一些示例文本。
      这是一些示例文本。这是一些示例文本。
      这是一些示例文本。这是一些示例文本。
   </div>
</div>
<div class="media">
   <a class="pull-left" href="#">
      <img class="media-object" src="img/64.jpg"
      alt="媒体对象">
   </a>
   <div class="media-body">
      <h4 class="media-heading">媒体标题</h4>
      这是一些示例文本。这是一些示例文本。
```

```
        这是一些示例文本。这是一些示例文本。
        这是一些示例文本。这是一些示例文本。
        这是一些示例文本。这是一些示例文本。
        这是一些示例文本。这是一些示例文本。
        <div class="media">
           <a class="pull-left" href="#">
              <img class="media-object" src="img/64.jpg"
              alt="媒体对象">
           </a>
           <div class="media-body">
              <h4 class="media-heading">媒体标题</h4>
              这是一些示例文本。这是一些示例文本。
              这是一些示例文本。这是一些示例文本。
              这是一些示例文本。这是一些示例文本。
              这是一些示例文本。这是一些示例文本。
              这是一些示例文本。这是一些示例文本。
           </div>
        </div>
     </div>
</div>

</body>
</html>
```

结果如下所示：



媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。



媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。



媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。

让我们来看看下面这个有关媒体对象列表 .media-list 的实例：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 媒体对象列表</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
```

```
    <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="media-list">
   <li class="media">
      <a class="pull-left" href="#">
         <img class="media-object" src="img/64.jpg"
         alt="通用的占位符图像">
      </a>
      <div class="media-body">
         <h4 class="media-heading">媒体标题</h4>
         <p>这是一些示例文本。这是一些示例文本。
            这是一些示例文本。这是一些示例文本。
            这是一些示例文本。这是一些示例文本。
            这是一些示例文本。这是一些示例文本。
            这是一些示例文本。这是一些示例文本。</p>
         <!-- 嵌套的媒体对象 -->
         <div class="media">
            <a class="pull-left" href="#">
               <img class="media-object" src="img/64.jpg"
               alt="通用的占位符图像">
            </a>
            <div class="media-body">
               <h4 class="media-heading">嵌套的媒体标题</h4>
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               <!-- 嵌套的媒体对象 -->
               <div class="media">
                  <a class="pull-left" href="#">
                     <img class="media-object" src="img/64.jpg"
                     alt="通用的占位符图像">
                  </a>
                  <div class="media-body">
                     <h4 class="media-heading">嵌套的媒体标题</h4>
                     这是一些示例文本。这是一些示例文本。
                     这是一些示例文本。这是一些示例文本。
                     这是一些示例文本。这是一些示例文本。
                     这是一些示例文本。这是一些示例文本。
                     这是一些示例文本。这是一些示例文本。
                  </div>
               </div>
            </div>
         </div>
         <!-- 嵌套的媒体对象 -->
         <div class="media">
            <a class="pull-left" href="#">
               <img class="media-object" src="img/64.jpg"
               alt="通用的占位符图像">
            </a>
```

```
            <div class="media-body">
               <h4 class="media-heading">嵌套的媒体标题</h4>
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
               这是一些示例文本。这是一些示例文本。
            </div>
         </div>
      </div>
   </li>
   <li class="media">
      <a class="pull-right" href="#">
         <img class="media-object" src="img/64.jpg"
         alt="通用的占位符图像">
      </a>
      <div class="media-body">
         <h4 class="media-heading">媒体标题</h4>
         这是一些示例文本。这是一些示例文本。
         这是一些示例文本。这是一些示例文本。
         这是一些示例文本。这是一些示例文本。
         这是一些示例文本。这是一些示例文本。
         这是一些示例文本。这是一些示例文本。
      </div>
   </li>
</ul>

</body>
</html>
```

结果如下所示：



媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。



嵌套的媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。



嵌套的媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。



嵌套的媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。

媒体标题

这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。 这是一些示例文本。这是一些示例文本。



Bootstrap 列表组

本章我们将讲解列表组。列表组件用于以列表形式呈现复杂的和自定义的内容。创建一个基本的列表组的步骤如下：

- 向元素 <ul> 添加 class **.list-group**。
- 向 <li> 添加 class **.list-group-item**。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 基本的列表组</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="list-group">
   <li class="list-group-item">免费域名注册</li>
   <li class="list-group-item">免费 Window 空间托管</li>
   <li class="list-group-item">图像的数量</li>
   <li class="list-group-item">24*7 支持</li>
   <li class="list-group-item">每年更新成本</li>
</ul>

</body>
</html>
```

结果如下所示：

免费域名注册
免费 Window 空间托管
图像的数量
24*7 支持
每年更新成本

向列表组添加徽章

我们可以向任意的列表项添加徽章组件，它会自动定位到右边。只需要在 <li> 元素中添加 **<span class="badge">** 即可。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 向列表组添加徽章</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul class="list-group">
   <li class="list-group-item">免费域名注册</li>
   <li class="list-group-item">免费 Window 空间托管</li>
   <li class="list-group-item">图像的数量</li>
   <li class="list-group-item">
      <span class="badge">新</span>
      24*7 支持
   </li>
   <li class="list-group-item">每年更新成本</li>
   <li class="list-group-item">
      <span class="badge">新</span>
      折扣优惠
   </li>
</ul>

</body>
</html>
```

结果如下所示：



向列表组添加链接

通过使用锚标签代替列表项，我们可以向列表组添加链接。我们需要使用 <div> 代替 <ul> 元素。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 向列表组添加链接</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<a href="#" class="list-group-item active">
   免费域名注册
</a>
<a href="#" class="list-group-item">24*7 支持</a>
<a href="#" class="list-group-item">免费 Window 空间托管</a>
<a href="#" class="list-group-item">图像的数量</a>
<a href="#" class="list-group-item">每年更新成本</a>

</body>
</html>
```

结果如下所示：



向列表组添加自定义内容

我们可以向上面已添加链接的列表组添加任意的 HTML 内容。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 向列表组添加自定义内容</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

```
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="list-group">
   <a href="#" class="list-group-item active">
      <h4 class="list-group-item-heading">
         入门网站包
      </h4>
   </a>
   <a href="#" class="list-group-item">
      <h4 class="list-group-item-heading">
         免费域名注册
      </h4>
      <p class="list-group-item-text">
         您将通过网页进行免费域名注册。
      </p>
   </a>
   <a href="#" class="list-group-item">
      <h4 class="list-group-item-heading">
         24*7 支持
      </h4>
      <p class="list-group-item-text">
         我们提供 24*7 支持。
      </p>
   </a>
</div>
<div class="list-group">
   <a href="#" class="list-group-item active">
      <h4 class="list-group-item-heading">
         商务网站包
      </h4>
   </a>
   <a href="#" class="list-group-item">
      <h4 class="list-group-item-heading">
         免费域名注册
      </h4>
      <p class="list-group-item-text">
         您将通过网页进行免费域名注册。
      </p>
   </a>
   <a href="#" class="list-group-item">
      <h4 class="list-group-item-heading">24*7 支持</h4>
      <p class="list-group-item-text">我们提供 24*7 支持。</p>
   </a>
</div>

</body>
</html>
```

结果如下所示：


入门网站包
免费域名注册
您将通过网页进行免费域名注册。
24*7 支持
我们提供 24*7 支持。
商务网站包
免费域名注册
您将通过网页进行免费域名注册。
24*7 支持
我们提供 24*7 支持。

Bootstrap 面板（Panels）

本章将讲解 Bootstrap 面板（Panels）。面板组件用于把 DOM 组件插入到一个盒子中。创建一个基本的面板，只需要向 <div> 元素添加 class **.panel** 和 class **.panel-default** 即可，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的面板</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="panel panel-default">
   <div class="panel-body">
      这是一个基本的面板
   </div>
</div>

</body>
</html>
```

结果如下所示：

这是一个基本的面板

面板标题

我们可以通过以下两种方式来添加面板标题：

- 使用 **.panel-heading** class 可以很简单地向面板添加标题容器。to easily add a heading container to your panel.
- 使用带有 **.panel-title** class 的 <h1>-<h6> 来添加预定义样式的标题。

下面的实例演示了这两种方式：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 面板标题</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="panel panel-default">
   <div class="panel-heading">
      不带 title 的面板标题
   </div>
   <div class="panel-body">
      面板内容
   </div>
</div>

<div class="panel panel-default">
   <div class="panel-heading">
      <h3 class="panel-title">
         带有 title 的面板标题
      </h3>
   </div>
   <div class="panel-body">
      面板内容
   </div>
</div>

</body>
</html>
```

结果如下所示：

不带 title 的面板标题

面板内容

带有 title 的面板标题

面板内容

面板脚注

我们可以在面板中添加脚注，只需要把按钮或者副文本放在带有 class **.panel-footer** 的 <div> 中即可。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 面板脚注</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="panel panel-default">
   <div class="panel-body">
      这是一个基本的面板
   </div>
   <div class="panel-footer">面板脚注</div>
</div>

</body>
</html>
```

结果如下所示：

这是一个基本的面板

面板脚注

> 面版脚注不会从带语境色彩的面板中继承颜色和边框，因为它不是前景中的内容。

带语境色彩的面板

使用语境状态类 **panel-primary**、**panel-success**、**panel-info**、**panel-warning**、**panel-danger**，来设置带语境色彩的面板，实例如下：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 带语境色彩的面板</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
```

```
<div class="panel panel-primary">
   <div class="panel-heading">
      <h3 class="panel-title">面板标题</h3>
   </div>
   <div class="panel-body">
      这是一个基本的面板
   </div>
</div>
<div class="panel panel-success">
   <div class="panel-heading">
      <h3 class="panel-title">面板标题</h3>
   </div>
   <div class="panel-body">
      这是一个基本的面板
   </div>
</div>
<div class="panel panel-info">
   <div class="panel-heading">
      <h3 class="panel-title">面板标题</h3>
   </div>
   <div class="panel-body">
      这是一个基本的面板
   </div>
</div>
<div class="panel panel-warning">
   <div class="panel-heading">
      <h3 class="panel-title">面板标题</h3>
   </div>
   <div class="panel-body">
      这是一个基本的面板
   </div>
</div>
<div class="panel panel-danger">
   <div class="panel-heading">
      <h3 class="panel-title">面板标题</h3>
   </div>
   <div class="panel-body">
      这是一个基本的面板
   </div>
</div>

</body>
</html>
```

结果如下所示：



带表格的面板

为了在面板中创建一个无边框的表格，我们可以在面板中使用 class **.table**。假设有个 <div> 包含 **.panel-body**，我们可以向表格的顶部添加额外的边框用来分隔。如果没有包含 **.panel-body** 的 <div>，则组件会无中断地从面板头部移动到表格。

下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 带表格的面板</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="panel panel-default">
   <div class="panel-heading">
      <h3 class="panel-title">面板标题</h3>
   </div>
   <div class="panel-body">
      这是一个基本的面板
   </div>
   <table class="table">
      <th>产品</th><th>价格 </th>
      <tr><td>产品 A</td><td>200</td></tr>
      <tr><td>产品 B</td><td>400</td></tr>
   </table>
</div>
<div class="panel panel-default">
   <div class="panel-heading">面板标题</div>
   <table class="table">
      <th>产品</th><th>价格 </th>
      <tr><td>产品 A</td><td>200</td></tr>
      <tr><td>产品 B</td><td>400</td></tr>
   </table>
</div>

</body>
</html>
```

结果如下所示：

面板标题

这是一个基本的面板

产品	价格
产品 A	200
产品 B	400

面板标题

产品	价格
产品 A	200
产品 B	400

带列表组的面板

我们可以在任何面板中包含列表组，通过在 <div> 元素中添加 **.panel** 和 **.panel-default** 类来创建面板，并在面板中添加列表组。您可以从 列表组 一章中学习如何创建列表组。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 带列表组的面板</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="panel panel-default">
   <div class="panel-heading">面板标题</div>
      <div class="panel-body">
         <p>这是一个基本的面板内容。这是一个基本的面板内容。
         这是一个基本的面板内容。这是一个基本的面板内容。
         这是一个基本的面板内容。这是一个基本的面板内容。
         这是一个基本的面板内容。这是一个基本的面板内容。
         </p>
   </div>
   <ul class="list-group">
      <li class="list-group-item">免费域名注册</li>
      <li class="list-group-item">免费 Window 空间托管</li>
      <li class="list-group-item">图像的数量</li>
      <li class="list-group-item">24*7 支持</li>
      <li class="list-group-item">每年更新成本</li>
   </ul>
</div>

</body>
</html>
```

结果如下所示：

面板标题

这是一个基本的面板内容。这是一个基本的面板内容。 这是一个基本的面板内容。这是一个基本的面板内容。 这是一个基本的面板内容。这是一个基本的面板内容。 这是一个基本的面板内容。这是一个基本的面板内容。

免费域名注册

免费 Window 空间托管

图像的数量

24*7 支持

每年更新成本

Bootstrap Wells

Well 是一种会引起内容凹陷显示或插图效果的容器 <div>。为了创建 Well，只需要简单地把内容放在带有 class **.well** 的 <div> 中即可。下面的实例演示了一个默认的 Well：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的 Well</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

 <div class="well">您好，我在 Well 中！</div>

</body>
</html>
```

结果如下所示：

您好，我在 Well 中！

尺寸大小

您可以使用可选的 class **well-lg** or **well-lg** 来改变 Well 的尺寸大小。这两个类是与 **.well** 类结合使用的。这两个类会影响内边距（padding），根据使用的类，Well 会显示得更大或者更小。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - Well 的尺寸大小</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="well well-lg">您好，我在大的 Well 中！</div>
<div class="well well-sm">您好，我在小的 Well 中！</div>

</body>
</html>
```

结果如下所示：

您好，我在大的 Well 中!

您好，我在小的 Well 中!

Bootstrap 插件

Bootstrap 插件概览

在前面 布局组件 章节中所讨论到的组件仅仅是个开始。Bootstrap 自带 12 种 jQuery 插件，扩展了功能，可以给站点添加更多的互动。即使您不是一名高级的 JavaScript 开发人员，您也可以着手学习 Bootstrap 的 JavaScript 插件。利用 Bootstrap 数据 API（Bootstrap Data API），大部分的插件可以在不编写任何代码的情况被触发。

站点引用 Bootstrap 插件的方式有两种：

- 单独引用：使用 Bootstrap 的个别的 **.js* 文件。一些插件和 CSS 组件依赖于其他插件。如果您单独引用插件，请先确保弄清这些插件之间的依赖关系。
- 编译（同时）引用：使用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

> 不要尝试同时引用这两个文件，因为 *bootstrap.js* 和 *bootstrap.min.js* 都包含了所有的插件。

所有的插件依赖于 ***jQuery***。所以必须在插件文件之前引用 ***jQuery***。请访问 ***bower.json*** 查看 ***Bootstrap*** 当前支持的 ***jQuery*** 版本。

data 属性

- 你可以仅仅通过 data 属性 API 就能使用所有的 Bootstrap 插件，无需写一行 JavaScript 代码。这是 Bootstrap 中的一等 API，也应该是你的首选方式。
- 话又说回来，在某些情况下可能需要将此功能关闭。因此，我们还提供了关闭 data 属性 API 的方法，即解除以 *data-api* 为命名空间并绑定在文档上的事件。就像下面这样：

```
$(document).off('.data-api')
```

- 如需关闭一个特定的插件，只需要在 data-api 命名空间前加上该插件的名称作为命名空间即可，如下所示：

```
$(document).off('.alert.data-api')
```

编程方式的 API

我们为所有 Bootstrap 插件提供了纯 JavaScript 方式的 API。所有公开的 API 都是支持单独或链式调用方式，并且返回其所操作的元素集合（注：和jQuery的调用形式一致）。例如：

```
$(".btn.danger").button("toggle").addClass("fat")
```

所有的方法都可以接受一个可选的选项对象作为参数，或者一个代表特定方法的字符串，或者不带任何参数（这种情况下，将会初始化插件为默认行为），如下所示：

```
// 初始化为默认行为
$("#myModal").modal()
 // 初始化为不支持键盘
$("#myModal").modal({ keyboard: false })
// 初始化并立即调用 show
$("#myModal").modal('show')
```

每个插件在 **Constructor** 属性上也暴露了其原始的构造函数：*$.fn.popover.Constructor*。如果您想获取某个特定插件的实例，可以直接通过页面元素获取：

```
 $('[rel=popover]').data('popover').
```

避免命名空间冲突

某些时候 Bootstrap 插件可能需要与其他 UI 框架一起使用。在这种情况下，可能会发生命名空间冲突。如果不幸发生了这种情况，你可以通过调用插件的 **.noConflict** 方法恢复其原始值。

```
// 返回 $.fn.button 之前所赋的值
var bootstrapButton = $.fn.button.noConflict()
// 为 $().bootstrapBtn 赋予 Bootstrap 功能
$.fn.bootstrapBtn = bootstrapButton
```

事件

Bootstrap 为大多数插件的独特行为提供了自定义事件。一般来说，这些事件有两种形式：

- 动词不定式：这会在事件开始时被触发。例如 *ex: show*。动词不定式事件提供了 *preventDefault* 功能。这使得在事件开始前可以停止操作的执行。

```
$('#myModal').on('show.bs.modal', function (e) {
// 阻止模态框的显示
  if (!data) return e.preventDefault()
})
```

- 过去分词形式：这会在动作执行完毕之后被触发。例如 *ex: shown*。

Bootstrap 过渡效果（Transition）插件

过渡效果（Transition）插件提供了简单的过渡效果。

> 如果您想要单独引用该插件的功能，那么除了其他的 JS 文件，您还需要引用 **transition.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

Transition.js 是 transitionEnd 事件和 CSS 过渡效果模拟器的基本帮助器类。它被其他插件用来检查 CSS 过渡效果支持，并用来获取过渡效果。

使用案例

过渡效果（Transition）插件的使用案例：

- 具有幻灯片或淡入效果的模态对话框。具体实例参见 Bootstrap 模态框（Modal）插件。
- 具有淡出效果的标签页。具体实例参见 Bootstrap 标签页（Tab）插件。
- 具有淡出效果的警告框。 具体实例参见 Bootstrap 警告框（Alert）插件。
- 具有幻灯片效果的轮播板。具体实例参见 Bootstrap 轮播（Carousel）插件。

Bootstrap 模态框（Modal）插件

模态框（Modal）是覆盖在父窗体上的子窗体。通常，目的是显示来自一个单独的源的内容，可以在不离开父窗体的情况下有一些互动。子窗体可提供信息、交互等。

> 如果您想要单独引用该插件的功能，那么您需要引用 **modal.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

您可以切换模态框（Modal）插件的隐藏内容：

- 通过 **data** 属性：在控制器元素（比如按钮或者链接）上设置属性 **data-toggle="modal"**，同时设置 **data-target="#identifier"** 或 **href="#identifier"** 来指定要切换的特定的模态框（带有 id="identifier"）。
- 通过 **JavaScript**：使用这种技术，您可以通过简单的一行 JavaScript 来调用带有 id="identifier" 的模态框：

```
$('#identifier').modal(options)
```

实例

一个静态的模态窗口实例，如下面的实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 模态框（Modal）插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h2>创建模态框（Modal）</h2>
<!-- 按钮触发模态框 -->
<button class="btn btn-primary btn-lg" data-toggle="modal"
   data-target="#myModal">
   开始演示模态框
</button>

<!-- 模态框（Modal） -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
   aria-labelledby="myModalLabel" aria-hidden="true">
   <div class="modal-dialog">
      <div class="modal-content">
         <div class="modal-header">
            <button type="button" class="close"
               data-dismiss="modal" aria-hidden="true">
                  &times;
            </button>
            <h4 class="modal-title" id="myModalLabel">
               模态框（Modal）标题
            </h4>
         </div>
         <div class="modal-body">
            在这里添加一些文本
         </div>
         <div class="modal-footer">
            <button type="button" class="btn btn-default"
               data-dismiss="modal">关闭
            </button>
            <button type="button" class="btn btn-primary">
               提交更改
            </button>
         </div>
      </div><!-- /.modal-content -->
</div><!-- /.modal -->

</body>
</html>
```

结果如下所示：



代码讲解：

- 使用模态窗口，您需要有某种触发器。您可以使用按钮或链接。这里我们使用的是按钮。
- 如果您仔细查看上面的代码，您会发现在 <button> 标签中，**data-target="#myModal"** 是您想要在页面上加载的模态框的目标。您可以在页面上创建多个模态框，然后为每个模态框创建不同的触发器。现在，很明显，您不能在同一时间加载多个模块，但您可以在页面上创建多个在不同时间进行加载。
- 在模态框中需要注意两点：
  - 第一是 **.modal**，用来把 <div> 的内容识别为模态框。
  - 第二是 **.fade** class。当模态框被切换时，它会引起内容淡入淡出。
- **aria-labelledby="myModalLabel"**，该属性引用模态框的标题。
- 属性 **aria-hidden="true"** 用于保持模态窗口不可见，直到触发器被触发为止（比如点击在相关的按钮上）。
- <div class="modal-header">，modal-header 是为模态窗口的头部定义样式的类。
- **class="close"**，close 是一个 CSS class，用于为模态窗口的关闭按钮设置样式。
- **data-dismiss="modal"**，是一个自定义的 HTML5 data 属性。在这里它被用于关闭模态窗口。
- **class="modal-body"**，是 Bootstrap CSS 的一个 CSS class，用于为模态窗口的主体设置样式。
- **class="modal-footer"**，是 Bootstrap CSS 的一个 CSS class，用于为模态窗口的底部设置样式。
- **data-toggle="modal"**，HTML5 自定义的 data 属性 data-toggle 用于打开模态窗口。

选项

有一些选项可以用来定制模态窗口（Modal Window）的外观和感观，它们是通过 data 属性或 JavaScript 来传递的。下表列出了这些选项：

选项名称	类型/默认值	Data 属性名称	
backdrop	boolean 或 string 'static' 默认值：*true*	data-backdrop	指定一个静态的背景，当用户点击模态框外
keyboard	boolean 默认值：*true*	data-keyboard	当按下 escape 键时关闭模态框，设置为 fals
show	boolean 默认值：*true*	data-show	当初始化时显示模态框。
remote	path 默认值：*false*	data-remote	使用 jQuery *.load* 方法，为模态框的主体注入内容。如下面的实例所示： `<a data-toggle="modal" href='`

方法

下面是一些可与 modal() 一起使用的有用的方法。

方法	描述	实例
Options: .modal(options)	把内容作为模态框激活。接受一个可选的选项对象。	`$('#identifier').modal({keyboard: false})`
Toggle: .modal('toggle')	手动切换模态框。	`$('#identifier').modal('toggle')`
Show: .modal('show')	手动打开模态框。	`$('#identifier').modal('show')`
Hide: .modal('hide')	手动隐藏模态框。	`$('#identifier').modal('hide')`

实例

下面的实例演示了方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 模态框（Modal）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h2>模态框（Modal）插件方法</h2>

<!-- 按钮触发模态框 -->
<button class="btn btn-primary btn-lg" data-toggle="modal" data-ta
   开始演示模态框
```

```
</button>

<!-- 模态框（Modal） -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
   aria-labelledby="myModalLabel" aria-hidden="true">
   <div class="modal-dialog">
      <div class="modal-content">
         <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal
               aria-hidden="true">×
            </button>
            <h4 class="modal-title" id="myModalLabel">
               模态框（Modal）标题
            </h4>
         </div>
         <div class="modal-body">
            按下 ESC 按钮退出。
         </div>
         <div class="modal-footer">
            <button type="button" class="btn btn-default"
               data-dismiss="modal">关闭
            </button>
            <button type="button" class="btn btn-primary">
               提交更改
            </button>
         </div>
      </div><!-- /.modal-content -->
   </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
<script>
   $(function () { $('#myModal').modal({
      keyboard: true
   })});
</script>

</body>
</html>
```

结果如下所示：



只需要点击 ESC 键，模态窗口即会退出。

事件

下表列出了模态框中要用到事件。这些事件可在函数中当钩子使用。

事件	描述	实例
show.bs.modal	在调用 show 方法后触发。	`$('#identifier').on('show.bs.modal', func`
shown.bs.modal	当模态框对用户可见时触发（将等待 CSS 过渡效果完成）。	`$('#identifier').on('shown.bs.modal', fun`
hide.bs.modal	当调用 hide 实例方法时触发。	`$('#identifier').on('hide.bs.modal', func`
hidden.bs.modal	当模态框完全对用户隐藏时触发。	`$('#identifier').on('hidden.bs.modal', fu`

实例

下面的实例演示了事件的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 模态框（Modal）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h2>模态框（Modal）插件事件</h2>
```

```
<!-- 按钮触发模态框 -->
<button class="btn btn-primary btn-lg" data-toggle="modal" data-tar
   开始演示模态框
</button>

<!-- 模态框（Modal） -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
   aria-labelledby="myModalLabel" aria-hidden="true">
   <div class="modal-dialog">
      <div class="modal-content">
         <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal
               aria-hidden="true">×
            </button>
            <h4 class="modal-title" id="myModalLabel">
               模态框（Modal）标题
            </h4>
         </div>
         <div class="modal-body">
            点击关闭按钮检查事件功能。
         </div>
         <div class="modal-footer">
            <button type="button" class="btn btn-default"
               data-dismiss="modal">
               关闭
            </button>
            <button type="button" class="btn btn-primary">
               提交更改
            </button>
         </div>
      </div><!-- /.modal-content -->
   </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
<script>
   $(function () { $('#myModal').modal('hide')})});
</script>
<script>
   $(function () { $('#myModal').on('hide.bs.modal', function () {
      alert('嘿，我听说您喜欢模态框...');})
   });
</script>

</body>
</html>
```

结果如下所示：



正如上面实例所示，如果您点击了 关闭 按钮，即 *hide* 事件，则会显示一个警告消息。

Bootstrap 下拉菜单（Dropdown）插件

Bootstrap 下拉菜单 这一章讲解了下拉菜单，但是没有涉及到交互部分，本章将具体讲解下拉菜单的交互。使用下拉菜单（Dropdown）插件，您可以向任何组件（比如导航栏、标签页、胶囊式导航菜单、按钮等）添加下拉菜单。

如果您想要单独引用该插件的功能，那么您需要引用 **dropdown.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

您可以切换下拉菜单（Dropdown）插件的隐藏内容：

- 通过 **data** 属性：向链接或按钮添加 **data-toggle="dropdown"** 来切换下拉菜单，如下所示：

```
&lt;div class="dropdown"&gt;
  &lt;a data-toggle="dropdown" href="#"&gt;下拉菜单（Dropdown）触
  &lt;ul class="dropdown-menu" role="menu" aria-labelledby="dLa
    ...
  &lt;/ul&gt;
&lt;/div&gt;
```

  如果您需要保持链接完整（在浏览器不启用 JavaScript 时有用），请使用 **data-target** 属性代替 **href="#"**：

```
&lt;div class="dropdown"&gt;
  &lt;a id="dLabel" role="button" data-toggle="dropdown" data-t
    下拉菜单（Dropdown） &lt;span class="caret"&gt;&lt;/span&gt;
  &lt;/a&gt;

  &lt;ul class="dropdown-menu" role="menu" aria-labelledby="dLa
    ...
  &lt;/ul&gt;
&lt;/div&gt;
```

- 通过 **JavaScript**：通过 JavaScript 调用下拉菜单切换，请使用下面的方法：

```
$('.dropdown-toggle').dropdown()
```

实例

在导航栏内

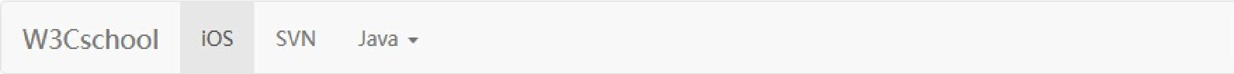
下面的实例演示了在导航栏内的下拉菜单的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 默认的导航栏</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>
   <div>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdo
               Java
               <b class="caret"></b>
            </a>
            <ul class="dropdown-menu">
               <li><a href="#">jmeter</a></li>
               <li><a href="#">EJB</a></li>
               <li><a href="#">Jasper Report</a></li>
               <li class="divider"></li>
               <li><a href="#">分离的链接</a></li>
               <li class="divider"></li>
               <li><a href="#">另一个分离的链接</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>

</body>
</html>
```

结果如下所示：



在标签页内

下面的实例演示了在标签页内的下拉菜单的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 带有下拉菜单的标签页</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<p>带有下拉菜单的标签页</p>
<ul class="nav nav-tabs">
   <li class="active"><a href="#">Home</a></li>
   <li><a href="#">SVN</a></li>
   <li><a href="#">iOS</a></li>
   <li><a href="#">VB.Net</a></li>
   <li class="dropdown">
      <a class="dropdown-toggle" data-toggle="dropdown" href="#">
         Java <span class="caret"></span>
      </a>
      <ul class="dropdown-menu">
         <li><a href="#">Swing</a></li>
         <li><a href="#">jMeter</a></li>
         <li><a href="#">EJB</a></li>
         <li class="divider"></li>
         <li><a href="#">分离的链接</a></li>
      </ul>
   </li>
   <li><a href="#">PHP</a></li>
</ul>

</body>
</html>
```

结果如下所示：



选项

没有选项。

方法

下拉菜单切换有一个简单的方法用来显示或隐藏下拉菜单。

```
$().dropdown('toggle')
```

实例

下面的实例演示了下拉菜单（Dropdown）插件方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 下拉菜单（Dropdown）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-default" role="navigation">
   <div class="navbar-header">
      <a class="navbar-brand" href="#">W3Cschool</a>
   </div>

   <div id="myexample">
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">iOS</a></li>
         <li><a href="#">SVN</a></li>
         <li class="dropdown">
            <a href="#" class="dropdown-toggle">Java <b
               class="caret"></b></a>
               <ul class="dropdown-menu">
                  <li><a id="action-1" href="#">
                     jmeter</a>
                  </li>
                  <li><a href="#">EJB</a></li>
                  <li><a href="#">Jasper Report</a></li>
                  <li class="divider"></li>
                  <li><a href="#">分离的链接</a></li>
                  <li class="divider"></li>
                  <li><a href="#">另一个分离的链接</a></li>
               </ul>
            </li>
         </ul>
      </div>
   </nav>
</div>
<script>
   $(function(){
      $(".dropdown-toggle").dropdown('toggle');
      });
</script>

</body>
</html>
```

结果如下所示：


W3Cschool
iOS
SVN
Java
jmeter
EJB
Jasper Report
分离的链接
另一个分离的链接

Bootstrap 滚动监听（Scrollspy）插件

滚动监听（Scrollspy）插件，即自动更新导航插件，会根据滚动条的位置自动更新对应的导航目标。其基本的实现是随着您的滚动，基于滚动条的位置向导航栏添加 **.active** class。

如果您想要单独引用该插件的功能，那么您需要引用 **scrollspy.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

您可以向顶部导航添加滚动监听行为：

- 通过 **data** 属性：向您想要监听的元素（通常是 body）添加 **data-spy="scroll"**。然后添加带有 Bootstrap **.nav** 组件的父元素的 ID 或 class 的属性 **data-target**。为了它能正常工作，您必须确保页面主体中有匹配您所要监听链接的 ID 的元素存在。

  ```
  &lt;body data-spy="scroll" data-target=".navbar-example"&gt;
    ...
    &lt;div class="navbar-example"&gt;
      &lt;ul class="nav nav-tabs"&gt;
        ...
      &lt;/ul&gt;
    &lt;/div&gt;
    ...
  &lt;/body&gt;
  ```

- 通过 **JavaScript**：您可以通过 JavaScript 调用滚动监听，选取要监听的元素，然后调用 **.scrollspy()** 函数：

  ```
  $('body').scrollspy({ target: '.navbar-example' })
  ```

实例

下面的实例演示了通过 data 属性使用滚动监听（Scrollspy）插件：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 滚动监听（Scrollspy）插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

```
    <script src="/scripts/jquery.min.js"></script>
    <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav id="navbar-example" class="navbar navbar-default navbar-static
    <div class="navbar-header">
        <button class="navbar-toggle" type="button" data-toggle="coll
            data-target=".bs-js-navbar-scrollspy">
            <span class="sr-only">切换导航</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="#">教程名称</a>
    </div>
    <div class="collapse navbar-collapse bs-js-navbar-scrollspy">
        <ul class="nav navbar-nav">
            <li><a href="#ios">iOS</a></li>
            <li><a href="#svn">SVN</a></li>
            <li class="dropdown">
                <a href="#" id="navbarDrop1" class="dropdown-toggle"
                    data-toggle="dropdown">Java
                    <b class="caret"></b>
                </a>
                <ul class="dropdown-menu" role="menu"
                    aria-labelledby="navbarDrop1">
                    <li><a href="#jmeter" tabindex="-1">jmeter</a></li>
                    <li><a href="#ejb" tabindex="-1">ejb</a></li>
                    <li class="divider"></li>
                    <li><a href="#spring" tabindex="-1">spring</a></li>
                </ul>
            </li>
        </ul>
    </div>
</nav>
<div data-spy="scroll" data-target="#navbar-example" data-offset="(
    style="height:200px;overflow:auto; position: relative;">
    <h4 id="ios">iOS</h4>
    <p>iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发布
        TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电
    </p>
    <h4 id="svn">SVN</h4>
    <p>Apache Subversion，通常缩写为 SVN，是一款开源的版本控制系统软件。Sub
    </p>
    <h4 id="jmeter">jMeter</h4>
    <p>jMeter 是一款开源的测试软件。它是 100% 纯 Java 应用程序，用于负载和性
    </p>
    <h4 id="ejb">EJB</h4>
    <p>Enterprise Java Beans（EJB）是一个创建高度可扩展性和强大企业级应用程
    </p>
    <h4 id="spring">Spring</h4>
    <p>Spring 框架是一个开源的 Java 平台，为快速开发功能强大的 Java 应用程序
```

```
    </p>
    <p>Spring 框架最初是由 Rod Johnson 编写的，在 2003 年 6 月首次发布于 A
    </p>
</div>

</body>
</html>
```

结果如下所示：



选项

通过 data 属性或 JavaScript 来传递。下表列出了这些选项：

选项名称	类型/默认值	Data 属性名称	描述
offset	number 默认值：*10*	data-offset	当计算滚动位置时，距离顶部的偏移像素。

方法

.scrollspy('refresh')：当通过 JavaScript 调用 scrollspy 方法时，您需要调用 **.refresh** 方法来更新 DOM。这在 DOM 的任意元素发生变更（即，您添加或移除了某些元素）时非常有用。下面是使用该方法的语法。

```
$('[data-spy="scroll"]').each(function () {
  var $spy = $(this).scrollspy('refresh')
})
```

实例

下面的实例演示了 **.scrollspy('refresh')** 方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 滚动监听（Scrollspy）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<nav id="myScrollspy" class="navbar navbar-default navbar-static" r
   <div class="navbar-header">
      <button class="navbar-toggle" type="button" data-toggle="coll
         data-target=".bs-js-navbar-scrollspy">
         <span class="sr-only">切换导航</span>
         <span class="icon-bar"></span>
         <span class="icon-bar"></span>
         <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">教程名称</a>
   </div>
   <div class="collapse navbar-collapse bs-js-navbar-scrollspy">
      <ul class="nav navbar-nav">
         <li class="active"><a href="#ios">iOS</a></li>
         <li><a href="#svn">SVN</a></li>
         <li class="dropdown">
            <a href="#" id="navbarDrop1" class="dropdown-toggle"
               data-toggle="dropdown">Java
               <b class="caret"></b>
            </a>
            <ul class="dropdown-menu" role="menu"
               aria-labelledby="navbarDrop1">
               <li><a href="#jmeter" tabindex="-1">jmeter</a></li>
               <li><a href="#ejb" tabindex="-1">ejb</a></li>
               <li class="divider"></li>
               <li><a href="#spring" tabindex="-1">spring</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>
<div data-spy="scroll" data-target="#myScrollspy" data-offset="0"
   style="height:200px;overflow:auto; position: relative;">
   <div class="section">
      <h4 id="ios">iOS<small><a href="#" onclick="removeSection(thi
         &times; 删除该部分</a></small>
      </h4>
      <p>iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发
```

```
        TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电
    </div>
    <div class="section">
        <h4 id="svn">SVN<small></small></h4>
        <p>Apache Subversion，通常缩写为 SVN，是一款开源的版本控制系统软件。
    </div>
    <div class="section">
        <h4 id="jmeter">jMeter<small><a href="#" onclick="removeSecti
        &times; 删除该部分</a></small>
        </h4>
        <p>jMeter 是一款开源的测试软件。它是 100% 纯 Java 应用程序，用于负载
    </div>
    <div class="section">
        <h4 id="ejb">EJB</h4>
        <p>Enterprise Java Beans（EJB）是一个创建高度可扩展性和强大企业级应
    </div>
    <div class="section">
        <h4 id="spring">Spring</h4>
        <p>Spring 框架是一个开源的 Java 平台，为快速开发功能强大的 Java 应用
        <p>Spring 框架最初是由 Rod Johnson 编写的，在 2003 年 6 月首次发布
    </div>
</div>
<script type="text/javascript">
    $(function(){
    removeSection = function(e) {
        $(e).parents(".section").remove();
        $('[data-spy="scroll"]').each(function () {
            var $spy = $(this).scrollspy('refresh')
        });
    }
    $("#myScrollspy").scrollspy();
});
</script>

</body>
</html>
```

结果如下所示：



事件

下表列出了滚动监听中要用到事件。这些事件可在函数中当钩子使用。

事件	描述	实
activate.bs.scrollspy	每当一个新项目被滚动监听激活时，触发该事件。	`$('#myScrollspy').on('activate.bs.scroll`

实例

下面的实例演示了 **activate.bs.scrollspy** 事件的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 滚动监听（Scrollspy）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
```

```
<nav id="myScrollspy" class="navbar navbar-default navbar-static" r
   <div class="navbar-header">
      <button class="navbar-toggle" type="button" data-toggle="coll
         data-target=".bs-js-navbar-scrollspy">
         <span class="sr-only">切换导航</span>
         <span class="icon-bar"></span>
         <span class="icon-bar"></span>
         <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">教程名称</a>
   </div>
   <div class="collapse navbar-collapse bs-js-navbar-scrollspy">
      <ul class="nav navbar-nav">
         <li class="active"><a href="#ios">iOS</a></li>
         <li><a href="#svn">SVN</a></li>
         <li class="dropdown">
            <a href="#" id="navbarDrop1" class="dropdown-toggle"
               data-toggle="dropdown">
               Java <b class="caret"></b>
            </a>
            <ul class="dropdown-menu" role="menu"
               aria-labelledby="navbarDrop1">
               <li><a href="#jmeter" tabindex="-1">jmeter</a></li>
               <li><a href="#ejb" tabindex="-1">ejb</a></li>
               <li class="divider"></li>
               <li><a href="#spring" tabindex="-1">spring</a></li>
            </ul>
         </li>
      </ul>
   </div>
</nav>
<div data-spy="scroll" data-target="#myScrollspy" data-offset="0"
   style="height:200px;overflow:auto; position: relative;">
   <div class="section">
      <h4 id="ios">iOS<small><a href="#" onclick="removeSection(thi
         &times; 删除该部分</a></small>
      </h4>
      <p>iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发
      TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电
   </div>
   <div class="section">
      <h4 id="svn">SVN<small></small></h4>
      <p>Apache Subversion，通常缩写为 SVN，是一款开源的版本控制系统软件。
   </div>
   <div class="section">
      <h4 id="jmeter">jMeter<small><a href="#" onclick="removeSecti
         &times; 删除该部分</a></small>
      </h4>
      <p>jMeter 是一款开源的测试软件。它是 100% 纯 Java 应用程序，用于负载
   </div>
   <div class="section">
      <h4 id="ejb">EJB</h4>
      <p>Enterprise Java Beans（EJB）是一个创建高度可扩展性和强大企业级应
```

```
    </div>
    <div class="section">
       <h4 id="spring">Spring</h4>
       <p>Spring 框架是一个开源的 Java 平台，为快速开发功能强大的 Java 应用
       <p>Spring 框架最初是由 Rod Johnson 编写的，在 2003 年 6 月首次发布
    </div>
 </div>
 <script type="text/javascript">
    $(function(){
       removeSection = function(e) {
       $(e).parents(".section").remove();
       $('[data-spy="scroll"]').each(function () {
           var $spy = $(this).scrollspy('refresh')
       });
    }
    $("#myScrollspy").scrollspy();
    $('#myScrollspy').on('activate.bs.scrollspy', function () {
       var currentItem = $(".nav li.active > a").text();
       $("#activeitem").html("目前您正在查看 - " + currentItem);
    })
 });
 </script>

 </body>
 </html>
```

结果如下所示：



Bootstrap 标签页（Tab）插件

标签页（Tab）在 Bootstrap 导航元素 一章中介绍过。通过结合一些 data 属性，您可以轻松地创建一个标签页界面。通过这个插件您可以把内容放置在标签页或者是胶囊式标签页甚至是下拉菜单标签页中。

> 如果您想要单独引用该插件的功能，那么您需要引用 **tab.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

您可以通过以下两种方式启用标签页：

- 通过 **data 属性**：您需要添加 **data-toggle="tab"** 或 **data-toggle="pill"** 到锚文本链接中。

  添加 **nav** 和 **nav-tabs** 类到 **ul** 中，将会应用 Bootstrap 标签样式，添加 **nav** 和 **nav-pills** 类到 **ul** 中，将会应用 Bootstrap 胶囊式样式。

  ```
  &lt;ul class="nav nav-tabs"&gt;
     &lt;li&gt;&lt;a href="#identifier" data-toggle="tab"&gt;Home
  ...
  &lt;/ul&gt;
  ```

- 通过 **JavaScript**：您可以使用 Javscript 来启用标签页，如下所示：

  ```
  $('#myTab a').click(function (e) {
    e.preventDefault()
    $(this).tab('show')
  })
  ```

  下面的实例演示了以不同的方式来激活各个标签页：

```
// 通过名称选取标签页
$('#myTab a[href="#profile"]').tab('show')

// 选取第一个标签页
$('#myTab a:first').tab('show')

// 选取最后一个标签页
$('#myTab a:last').tab('show')

// 选取第三个标签页（从 0 开始索引）
$('#myTab li:eq(2) a').tab('show')
```

淡入淡出效果

如果需要为标签页设置淡入淡出效果，请添加 **.fade** 到每个 **.tab-pane** 后面。第一个标签页必须添加 **.in** 类，以便淡入显示初始内容，如下面实例所示：

```
<div class="tab-content">
  <div class="tab-pane fade in active" id="home">...</div>
  <div class="tab-pane fade" id="svn">...</div>
  <div class="tab-pane fade" id="ios">...</div>
  <div class="tab-pane fade" id="java">...</div>
</div>
```

实例

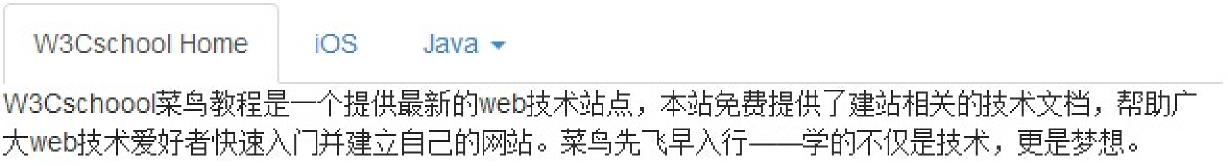
下面的实例演示了使用 data 属性的标签页（Tab）插件及其淡入淡出的效果：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标签页（Tab）插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul id="myTab" class="nav nav-tabs">
   <li class="active">
      <a href="#home" data-toggle="tab">
         W3Cschool Home
      </a>
   </li>
   <li><a href="#ios" data-toggle="tab">iOS</a></li>
   <li class="dropdown">
      <a href="#" id="myTabDrop1" class="dropdown-toggle"
         data-toggle="dropdown">Java
         <b class="caret"></b>
      </a>
      <ul class="dropdown-menu" role="menu" aria-labelledby="myTabD
         <li><a href="#jmeter" tabindex="-1" data-toggle="tab">jmet
         <li><a href="#ejb" tabindex="-1" data-toggle="tab">ejb</a>
      </ul>
   </li>
</ul>
<div id="myTabContent" class="tab-content">
   <div class="tab-pane fade in active" id="home">
      <p>W3Cschoool菜鸟教程是一个提供最新的web技术站点，本站免费提供了建站相
   </div>
   <div class="tab-pane fade" id="ios">
      <p>iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发
      TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电
   </div>
   <div class="tab-pane fade" id="jmeter">
      <p>jMeter 是一款开源的测试软件。它是 100% 纯 Java 应用程序，用于负载
   </div>
   <div class="tab-pane fade" id="ejb">
      <p>Enterprise Java Beans（EJB）是一个创建高度可扩展性和强大企业级应
      </p>
   </div>
</div>

</body>
</html>
```

结果如下所示：



方法

.$().tab：该方法可以激活标签页元素和内容容器。标签页需要用一个 **data-target** 或者一个指向 DOM 中容器节点的 **href**。

```
<ul class="nav nav-tabs" id="myTab">
  <li class="active"><a href="#identifier" data-toggle="tab">Home</
  .....
</ul>
<div class="tab-content">
  <div class="tab-pane active" id="home">...</div>
  .....
</div>
<script>
  $(function () {
    $('#myTab a:last').tab('show')
  })
</script>
```

实例

下面的实例演示了标签页（Tab）插件方法 **.tab** 的用法。在本实例中，第二个标签页 *iOS* 是激活的：

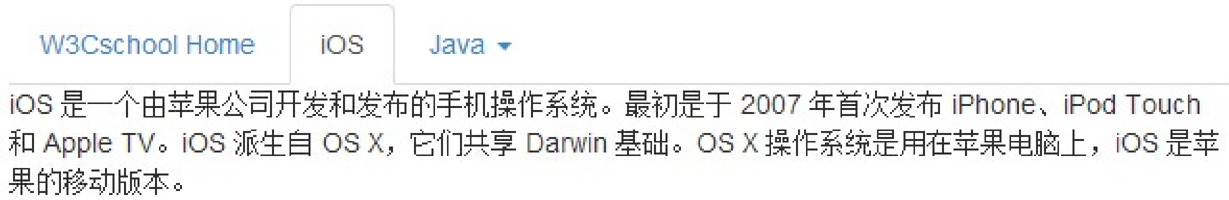
```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标签页（Tab）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<ul id="myTab" class="nav nav-tabs">
   <li class="active"><a href="#home" data-toggle="tab">
      W3Cschool Home</a>
   </li>
   <li><a href="#ios" data-toggle="tab">iOS</a></li>
   <li class="dropdown">
      <a href="#" id="myTabDrop1" class="dropdown-toggle"
```

```
            data-toggle="dropdown">Java <b class="caret"></b>
         </a>
         <ul class="dropdown-menu" role="menu" aria-labelledby="myTabD
            <li><a href="#jmeter" tabindex="-1" data-toggle="tab">
               jmeter</a>
            </li>
            <li><a href="#ejb" tabindex="-1" data-toggle="tab">
               ejb</a>
            </li>
         </ul>
      </li>
   </ul>
   <div id="myTabContent" class="tab-content">
      <div class="tab-pane fade in active" id="home">
         <p>W3Cschoool菜鸟教程是一个提供最新的web技术站点，本站免费提供了建站相
      </div>
      <div class="tab-pane fade" id="ios">
         <p>iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发
         TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电
      </div>
      <div class="tab-pane fade" id="jmeter">
         <p>jMeter 是一款开源的测试软件。它是 100% 纯 Java 应用程序，用于负载
      </div>
      <div class="tab-pane fade" id="ejb">
         <p>Enterprise Java Beans（EJB）是一个创建高度可扩展性和强大企业级应
         </p>
      </div>
   </div>
   <script>
      $(function () {
         $('#myTab li:eq(1) a').tab('show');
      });
   </script>

   </body>
   </html>
```

结果如下所示：



事件

下表列出了标签页（Tab）插件中要用到的事件。这些事件可在函数中当钩子使用。

事件	描述	
show.bs.tab	该事件在标签页显示时触发，但是必须在新标签页被显示之前。分别使用 **event.target** 和 **event.relatedTarget** 来定位到激活的标签页和前一个激活的标签页。	`$('a[data-toggle="tab"]').on('sh`
shown.bs.tab	该事件在标签页显示时触发，但是必须在某个标签页已经显示之后。分别使用 **event.target** 和 **event.relatedTarget** 来定位到激活的标签页和前一个激活的标签页。	`$('a[data-toggle="tab"]').on('sh`

实例

下面的实例演示了标签页（Tab）插件事件的用法。在本实例中，将显示当前和前一个访问过的标签页：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 标签页（Tab）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<hr>
   <p class="active-tab"><strong>激活的标签页</strong>：<span></span>
   <p class="previous-tab"><strong>前一个激活的标签页</strong>：<span>
<hr>
<ul id="myTab" class="nav nav-tabs">
   <li class="active"><a href="#home" data-toggle="tab">
      W3Cschool Home</a></li>
   <li><a href="#ios" data-toggle="tab">iOS</a></li>
   <li class="dropdown">
```

```
        <a href="#" id="myTabDrop1" class="dropdown-toggle"
           data-toggle="dropdown">
           Java <b class="caret"></b></a>
        <ul class="dropdown-menu" role="menu" aria-labelledby="myTabD
           <li><a href="#jmeter" tabindex="-1" data-toggle="tab">jmet
           <li><a href="#ejb" tabindex="-1" data-toggle="tab">ejb</a>
        </ul>
     </li>
  </ul>
  <div id="myTabContent" class="tab-content">
     <div class="tab-pane fade in active" id="home">
        <p>W3Cschoool菜鸟教程是一个提供最新的web技术站点，本站免费提供了建站相
     </div>
     <div class="tab-pane fade" id="ios">
        <p>iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发
        TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电
     </div>
     <div class="tab-pane fade" id="jmeter">
        <p>jMeter 是一款开源的测试软件。它是 100% 纯 Java 应用程序，用于负载
     </div>
     <div class="tab-pane fade" id="ejb">
        <p>Enterprise Java Beans（EJB）是一个创建高度可扩展性和强大企业级应
        </p>
     </div>
  </div>
  <script>
     $(function(){
        $('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {
        // 获取已激活的标签页的名称
        var activeTab = $(e.target).text();
        // 获取前一个激活的标签页的名称
        var previousTab = $(e.relatedTarget).text();
        $(".active-tab span").html(activeTab);
        $(".previous-tab span").html(previousTab);
     });
  });
  </script>

  </body>
  </html>
```

结果如下所示：

激活的标签页： iOS

前一个激活的标签页： W3Cschool Home

W3Cschool Home | iOS | Java ▾

iOS 是一个由苹果公司开发和发布的手机操作系统。最初是于 2007 年首次发布 iPhone、iPod Touch 和 Apple TV。iOS 派生自 OS X，它们共享 Darwin 基础。OS X 操作系统是用在苹果电脑上，iOS 是苹果的移动版本。

Bootstrap 工具提示（Tooltip）插件

当您想要描述一个链接的时候，工具提示（Tooltip）就显得非常有用。工具提示（Tooltip）插件是受 *Jason Frame* 写的 *jQuery.tipsy* 的启发。工具提示（Tooltip）插件做了很多改进，例如不需要依赖图像，而是改用 CSS 实现动画效果，用 data 属性存储标题信息。

> 如果您想要单独引用该插件的功能，那么您需要引用 **tooltip.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

工具提示（Tooltip）插件根据需求生成内容和标记，默认情况下是把工具提示（tooltip）放在它们的触发元素后面。您可以有以下两种方式添加工具提示（tooltip）：

- 通过 **data 属性**：如需添加一个工具提示（tooltip），只需向一个锚标签添加 **data-toggle="tooltip"** 即可。锚的 title 即为工具提示（tooltip）的文本。默认情况下，插件把工具提示（tooltip）设置在顶部。

  ```
  &lt;a href="#" data-toggle="tooltip" title="Example tooltip"&gt
  ```

- 通过 **JavaScript**：通过 JavaScript 触发工具提示（tooltip）：

  ```
  $('#identifier').tooltip(options)
  ```

> 工具提示（Tooltip）插件不像之前所讨论的下拉菜单及其他插件那样，它不是纯 CSS 插件。如需使用该插件，您必须使用 jquery 激活它（读取 javascript）。使用下面的脚本来启用页面中的所有的工具提示（tooltip）：
>
> ```
> $(function () { $("[data-toggle='tooltip']").tooltip(); });
> ```

实例

下面的实例演示了通过 data 属性使用工具提示（Tooltip）插件的用法。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 工具提示（Tooltip）插件</title>
```

```
    <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
    <script src="/scripts/jquery.min.js"></script>
    <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h4>工具提示（Tooltip）插件 - 锚</h4>
这是一个 <a href="#" class="tooltip-test" data-toggle="tooltip"
    title="默认的 Tooltip">
    默认的 Tooltip
</a>.
这是一个 <a href="#" class="tooltip-test" data-toggle="tooltip"
    data-placement="left" title="左侧的 Tooltip">
    左侧的 Tooltip
</a>.
这是一个 <a href="#" data-toggle="tooltip" data-placement="top"
    title="顶部的 Tooltip">
    顶部的 Tooltip
</a>.
这是一个 <a href="#" data-toggle="tooltip" data-placement="bottom"
    title="底部的 Tooltip">
    底部的 Tooltip
</a>.
这是一个 <a href="#" data-toggle="tooltip" data-placement="right"
    title="右侧的 Tooltip">
    右侧的 Tooltip
</a>

<br>

<h4>工具提示（Tooltip）插件 - 按钮</h4>
<button type="button" class="btn btn-default" data-toggle="tooltip"
    title="默认的 Tooltip">
    默认的 Tooltip
</button>
<button type="button" class="btn btn-default" data-toggle="tooltip"
    data-placement="left" title="左侧的 Tooltip">
    左侧的 Tooltip
</button>
<button type="button" class="btn btn-default" data-toggle="tooltip"
    data-placement="top" title="顶部的 Tooltip">
    顶部的 Tooltip
</button>
<button type="button" class="btn btn-default" data-toggle="tooltip"
    data-placement="bottom" title="底部的 Tooltip">
    底部的 Tooltip
</button>
<button type="button" class="btn btn-default" data-toggle="tooltip"
    data-placement="right" title="右侧的 Tooltip">
    右侧的 Tooltip
</button>
<script>
    $(function () { $("[data-toggle='tooltip']").tooltip(); });
```

```
</script>

</body>
</html>
```

结果如下所示：



选项

有一些选项是通过 Bootstrap 数据 API（Bootstrap Data API）添加或通过 JavaScript 调用的。下表列出了这些选项：

选项名称	类型/默认值	**Data** 属性名称	描述
animation	boolean 默认值：*true*	data-animation	工具提示使用 CSS 渐变滤镜效果。
html	boolean 默认值：*false*	data-html	向工具提示插入 HTML。如果为 fals jQuery 的 text 方法将被用于向 dom 容。如果您担心 XSS 攻击，请使用
placement	string|function 默认值：*top*	data-placement	规定如何定位工具提示（即 top|bottom|left|right|auto）。当指定 时，会动态调整工具提示。例如，如 placement 是 "auto left"，工具提示 可能显示在左边，在情况不允许的情 才会显示在右边。
selector	string 默认值：*false*	data-selector	如果提供了一个选择器，工具提示对 委派到指定的目标。
title	string | function 默认值：*''*	data-title	如果未指定 *title* 属性，则 title 选项是 的 title 值。
trigger	string 默认值：*'hover focus'*	data-trigger	定义如何触发工具提示： **click| hov focus | manual**。您可以传递多个触 每个触发器之间用空格分隔。
content	string | function 默认值：*''*	data-content	如果未指定 *data-content* 属性，则使 认的 content 值。
delay	number | object 默认值：*0*	data-delay	延迟显示和隐藏工具提示的毫秒数 - manual 手动触发类型不适用。如果 是一个数字，那么延迟将会应用于显 藏。如果提供的是对象，结构如下所 示： `delay:{ show: 500, hide:`
container	string | false 默认值：*false*	data-container	向指定元素追加工具提示。实例： container: 'body'

方法

下面是一些工具提示（Tooltip）插件中有用的方法：

方法	描述	实例
Options: .tooltip(options)	向元素集合附加提示工具句柄。	`$().tooltip(options)`
Toggle: .tooltip('toggle')	切换显示/隐藏元素的提示工具。	`$('#element').tooltip('toggle')`
Show: .tooltip('show')	显示元素的提示工具。	`$('#element').tooltip('show')`
Hide: .tooltip('hide')	隐藏元素的提示工具。	`$('#element').tooltip('hide')`
Destroy: .tooltip('destroy')	隐藏并销毁元素的提示工具。	`$('#element').tooltip('destroy')`

实例

下面的实例演示了工具提示（Tooltip）插件方法的用法。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 工具提示（Tooltip）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div style="padding: 100px 100px 10px;">
   这是一个 <a href="#" class="tooltip-show" data-toggle="tooltip"
      title="show">Tooltip 方法 show
   </a>.

   这是一个 <a href="#" class="tooltip-hide" data-toggle="tooltip"
      data-placement="left" title="hide">Tooltip 方法 hide
   </a>.

   这是一个 <a href="#" class="tooltip-destroy" data-toggle="tooltip
      data-placement="top" title="destroy">Tooltip 方法 destroy
   </a>.

   这是一个 <a href="#" class="tooltip-toggle" data-toggle="tooltip"
      data-placement="bottom" title="toggle">Tooltip 方法 toggle
   </a>.
   <br><br><br><br><br><br>
   <p class="tooltip-options" >
      这是一个 <a href="#" data-toggle="tooltip" title="<h2>'am Head
         </h2>">Tooltip 方法 options
      </a>.
   </p>

   <script>
      $(function () { $('.tooltip-show').tooltip('show');});
      $(function () { $('.tooltip-hide').tooltip('hide');});
      $(function () { $('.tooltip-destroy').tooltip('destroy');});
      $(function () { $('.tooltip-toggle').tooltip('toggle');});
      $(function () { $(".tooltip-options a").tooltip({html : true
      });
   </script>
<div>

</body>
</html>
```

结果如下所示：



事件

下表列出了工具提示（Tooltip）插件中要用到的事件。这些事件可在函数中当钩子使用。

事件	描述	实例
show.bs.tooltip	当调用 show 实例方法时立即触发该事件。	`$('#myTooltip').on('show.bs.tooltip', fun`
shown.bs.tooltip	当工具提示对用户可见时触发该事件（将等待 CSS 过渡效果完成）。	`$('#myTooltip').on('shown.bs.tooltip', fu`
hide.bs.tooltip	当调用 hide 实例方法时立即触发该事件。	`$('#myTooltip').on('hide.bs.tooltip', fun`
hidden.bs.tooltip	当工具提示对用户隐藏时触发该事件（将等待 CSS 过渡效果完成）。	`$('#myTooltip').on('hidden.bs.tooltip', f`

实例

下面的实例演示了工具提示（Tooltip）插件事件的用法。

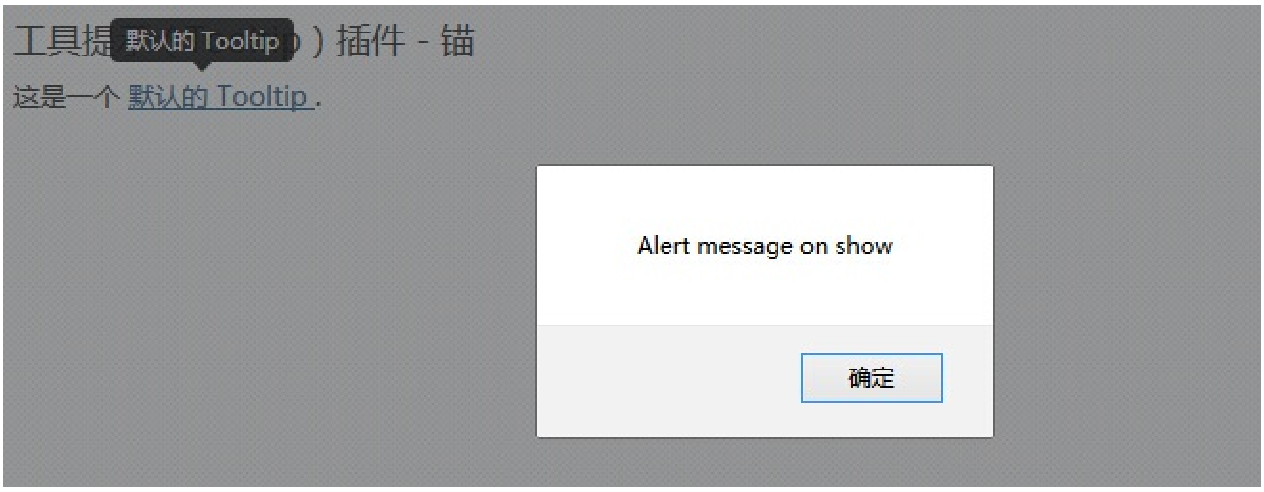
```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 工具提示（Tooltip）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h4>工具提示（Tooltip）插件 - 锚</h4>
这是一个 <a href="#" class="tooltip-show" data-toggle="tooltip"
   title="默认的 Tooltip">默认的 Tooltip
</a>.

<script>
$(function () { $('.tooltip-show').tooltip('show');});
$(function () { $('.tooltip-show').on('show.bs.tooltip', function (
   alert("Alert message on show");
})});
</script>

</body>
</html>
```

结果如下所示：



Bootstrap 弹出框（Popover）插件

弹出框（Popover）与工具提示（Tooltip）类似，提供了一个扩展的视图。如需激活弹出框，用户只需把鼠标悬停在元素上即可。弹出框的内容完全可使用 Bootstrap 数据 API（Bootstrap Data API）来填充。该方法依赖于工具提示（tooltip）。

如果您想要单独引用该插件的功能，那么您需要引用 **popover.js**，它依赖于 工具提示（Tooltip）插件。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

弹出框（Popover）插件根据需求生成内容和标记，默认情况下是把弹出框（popover）放在它们的触发元素后面。您可以有以下两种方式添加弹出框（popover）：

- 通过 **data** 属性：如需添加一个弹出框（popover），只需向一个锚/按钮标签添加 **data-toggle="popover"** 即可。锚的 title 即为弹出框（popover）的文本。默认情况下，插件把弹出框（popover）设置在顶部。

```
&lt;a href="#" data-toggle="popover" title="Example popover"&gt
   请悬停在我的上面
&lt;/a&gt;
```

- 通过 **JavaScript**：通过 JavaScript 启用弹出框（popover）：

```
$('#identifier').popover(options)
```

弹出框（Popover）插件不像之前所讨论的下拉菜单及其他插件那样，它不是纯 CSS 插件。如需使用该插件，您必须使用 jquery 激活它（读取 javascript）。使用下面的脚本来启用页面中的所有的弹出框（popover）：

```
$(function () { $("[data-toggle='popover']").popover(); });
```

实例

下面的实例演示了通过 data 属性使用弹出框（Popover）插件的用法。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 弹出框（Popover）插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container" style="padding: 100px 50px 10px;" >
   <button type="button" class="btn btn-default" title="Popover tit
      data-container="body" data-toggle="popover" data-placement="l
      data-content="左侧的 Popover 中的一些内容">
      左侧的 Popover
   </button>
   <button type="button" class="btn btn-primary" title="Popover tit
      data-container="body" data-toggle="popover" data-placement="t
      data-content="顶部的 Popover 中的一些内容">
      顶部的 Popover
   </button>
   <button type="button" class="btn btn-success" title="Popover tit
      data-container="body" data-toggle="popover" data-placement="b
      data-content="底部的 Popover 中的一些内容">
      底部的 Popover
   </button>
   <button type="button" class="btn btn-warning" title="Popover tit
      data-container="body" data-toggle="popover" data-placement="r
      data-content="右侧的 Popover 中的一些内容">
      右侧的 Popover
   </button>
   </div>

   <script>$(function ()
      { $("[data-toggle='popover']").popover();
      });
   </script>
</div>

</body>
</html>
```

结果如下所示：



选项

有一些选项是通过 Bootstrap 数据 API（Bootstrap Data API）添加或通过 JavaScript 调用的。下表列出了这些选项：

选项名称	类型/默认值	Data 属性名称	描述
animation	boolean 默认值：*true*	data-animation	向弹出框应用 CSS 褪色过渡效果。
html	boolean 默认值：*false*	data-html	向弹出框插入 HTML。如果为 false, jQuery 的 text 方法将被用于向 dom 容。如果您担心 XSS 攻击，请使用
placement	string|function 默认值：*top*	data-placement	规定如何定位弹出框（即 top|bottom|left|right|auto）。当指定时，会动态调整弹出框。例如，如果 placement 是 "auto left"，弹出框将 能显示在左边，在情况不允许的情况 会显示在右边。
selector	string 默认值：*false*	data-selector	如果提供了一个选择器，弹出框对象 派到指定的目标。
title	string | function 默认值：*''*	data-title	如果未指定 *title* 属性，则 title 选项是 的 title 值。
trigger	string 默认值：*'hover focus'*	data-trigger	定义如何触发弹出框： **click| hover focus | manual**。您可以传递多个触 每个触发器之间用空格分隔。
delay	number | object 默认值：*0*	data-delay	延迟显示和隐藏弹出框的毫秒数 - 对 manual 手动触发类型不适用。如果 是一个数字，那么延迟将会应用于显 藏。如果提供的是对象，结构如下所 示：`delay:{ show: 500, hide:`
container	string | false 默认值：*false*	data-container	向指定元素追加弹出框。实例：cor 'body'

方法

下面是一些弹出框（Popover）插件中有用的方法：

方法	描述	实例
Options: .popover(options)	向元素集合附加弹出框句柄。	`$().popover(options)`
Toggle: .popover('toggle')	切换显示/隐藏元素的弹出框。	`$('#element').popover('toggle')`
Show: .popover('show')	显示元素的弹出框。	`$('#element').popover('show')`
Hide: .popover('hide')	隐藏元素的弹出框。	`$('#element').popover('hide')`
Destroy: .popover('destroy')	隐藏并销毁元素的弹出框。	`$('#element').popover('destroy')`

实例

下面的实例演示了弹出框（Popover）插件的方法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 弹出框（Popover）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container" style="padding: 100px 50px 10px;" >
   <button type="button" class="btn btn-default popover-show"
      title="Popover title" data-container="body"
      data-toggle="popover" data-placement="left"
      data-content="左侧的 Popover 中的一些内容 —— show 方法">
      左侧的 Popover
   </button>
   <button type="button" class="btn btn-primary popover-hide"
      title="Popover title" data-container="body"
      data-toggle="popover" data-placement="top"
      data-content="顶部的 Popover 中的一些内容 —— hide 方法">
      顶部的 Popover
```

```
    </button>
    <button type="button" class="btn btn-success popover-destroy"
       title="Popover title" data-container="body"
       data-toggle="popover" data-placement="bottom"
       data-content="底部的 Popover 中的一些内容 —— destroy 方法">
       底部的 Popover
    </button>
    <button type="button" class="btn btn-warning popover-toggle"
       title="Popover title" data-container="body"
       data-toggle="popover" data-placement="right"
       data-content="右侧的 Popover 中的一些内容 —— toggle 方法">
       右侧的 Popover
    </button><br><br><br><br><br><br>
    <p class="popover-options">
       <a href="#" type="button" class="btn btn-warning" title="<h2>
          data-container="body" data-toggle="popover" data-content='
          <h4>Popover 中的一些内容 —— options 方法</h4>">
          Popover
       </a>
    </p>
    <script>
       $(function () { $('.popover-show').popover('show');});
       $(function () { $('.popover-hide').popover('hide');});
       $(function () { $('.popover-destroy').popover('destroy');});
       $(function () { $('.popover-toggle').popover('toggle');});
      $(function () { $(".popover-options a").popover({html : true ]
    </script>
 </div>

 </body>
 </html>
```

结果如下所示：



事件

下表列出了弹出框（Popover）插件中要用到的事件。这些事件可在函数中当钩子使用。

事件	描述	实例
show.bs.popover	当调用 show 实例方法时立即触发该事件。	`$('#mypopover').on('show.bs.popover', f`
shown.bs.popover	当弹出框对用户可见时触发该事件（将等待 CSS 过渡效果完成）。	`$('#mypopover').on('shown.bs.popover', `
hide.bs.popover	当调用 hide 实例方法时立即触发该事件。	`$('#mypopover').on('hide.bs.popover', f`
hidden.bs.popover	当工具提示对用户隐藏时触发该事件（将等待 CSS 过渡效果完成）。	`$('#mypopover').on('hidden.bs.popover',`

实例

下面的实例演示了弹出框（Popover）插件的事件：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 弹出框（Popover）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container" style="padding: 100px 50px 10px;" >
   <button type="button" class="btn btn-primary popover-show"
      title="Popover title" data-container="body"
      data-toggle="popover"
      data-content="左侧的 Popover 中的一些内容 —— show 方法">
      左侧的 Popover
   </button>

   </div>
   <script>
      $(function () { $('.popover-show').popover('show');});
      $(function () { $('.popover-show').on('shown.bs.popover', fur
      alert("当显示时警告消息");
   })});
   </script>
</div>

</body>
</html>
```

结果如下所示：



Bootstrap 警告框（Alert）插件

警告框（Alert）消息大多是用来想终端用户显示诸如警告或确认消息的信息。使用警告框（Alert）插件，您可以向所有的警告框消息添加可取消（dismiss）功能。

> 如果您想要单独引用该插件的功能，那么您需要引用 **alert.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

您可以有以下两种方式启用警告框的可取消（dismissal）功能：

- 通过 **data** 属性：通过数据 API（Data API）添加可取消功能，只需要向关闭按钮添加 **data-dismiss="alert"**，就会自动为警告框添加关闭功能。

```
&lt;a class="close" data-dismiss="alert" href="#" aria-hidden="
   &times;
&lt;/a&gt;
```

- 通过 **JavaScript**：通过 JavaScript 添加可取消功能：

```
$(".alert").alert()
```

实例

下面的实例演示了通过 data 属性使用警告框（Alert）插件的用法。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 警告框（Alert）插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="alert alert-warning">
   <a href="#" class="close" data-dismiss="alert">
      &times;
   </a>
   <strong>警告！</strong>您的网络连接有问题。
</div>

</body>
</html>
```

结果如下所示：

警告！ 您的网络连接有问题。 ×

选项

没有选项。

方法

下面是一些警告框（Alert）插件中有用的方法：

方法	描述	实例
.alert()	该方法让所有的警告框都带有关闭功能。	`$('#identifier').alert();`
Close Method .alert('close')	关闭所有的警告框。	`$('#identifier').alert('close');`

> 如需在关闭时启用动画效果，请确保添加了 **.fade** 和 **.in** class。

实例

下面的实例演示了 **.alert()** 方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 警告框（Alert）插件 alert() 方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h3>警告框（Alert）插件 alert() 方法</h3>
<div id="myAlert" class="alert alert-success">
   <a href="#" class="close" data-dismiss="alert">&times;</a>
   <strong>成功！</strong>结果是成功的。
</div>
<div id="myAlert" class="alert alert-warning">
   <a href="#" class="close" data-dismiss="alert">&times;</a>
   <strong>警告！</strong>您的网络连接有问题。
</div>

<script type="text/javascript">
$(function(){
   $(".close").click(function(){
      $("#myAlert").alert();
   });
});
</script>

</body>
</html>
```

下面的实例演示了 **.alert('close')** 方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 警告框（Alert）插件 alert('close') 方法</ti
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h3>警告框（Alert）插件 alert('close') 方法</h3>
<div id="myAlert" class="alert alert-success">
   <a href="#" class="close" data-dismiss="alert">&times;</a>
   <strong>成功！</strong>结果是成功的。
</div>
<div id="myAlert" class="alert alert-warning">
   <a href="#" class="close" data-dismiss="alert">&times;</a>
   <strong>警告！</strong>您的网络连接有问题。
</div>

<script type="text/javascript">
$(function(){
   $(".close").click(function(){
      $("#myAlert").alert('close');
   });
});

</script>

</body>
</html>
```

您可以看到所有的警告框都应用了关闭功能，即关闭任意的警告框，其他剩余的警告框也会被关闭。

事件

下表列出了警告框（Alert）插件中要用到的事件。这些事件可在函数中当钩子使用。

事件	描述	实例
close.bs.alert	当调用 *close* 实例方法时立即触发该事件。	`$('#myalert').bind('close.bs.alert', functi`
closed.bs.alert	当警告框被关闭时触发该事件（将等待 CSS 过渡效果完成）。	`$('#myalert').bind('closed.bs.alert', funct`

实例

下面的实例演示了警告框（Alert）插件的事件：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 警告框（Alert）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div id="myAlert" class="alert alert-success">
   <a href="#" class="close" data-dismiss="alert">&times;</a>
   <strong>成功！</strong>结果是成功的。
</div>

<script type="text/javascript">
$(function(){
   $("#myAlert").bind('closed.bs.alert', function () {
      alert("警告消息框被关闭。");
   });
});
</script>

</body>
</html>
```

结果如下所示：



Bootstrap 按钮（Button）插件

按钮（Button）在 Bootstrap 按钮 一章中介绍过。通过按钮（Button）插件，您可以添加进一些交互，比如控制按钮状态，或者为其他组件（如工具栏）创建按钮组。

> 如果您想要单独引用该插件的功能，那么您需要引用 **button.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

加载状态

如需向按钮添加加载状态，只需要简单地向 button 元素添加 **data-loading-text="Loading..."** 作为其属性即可，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮（Button）插件加载状态</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<button id="fat-btn" class="btn btn-primary" data-loading-text="Loa
   type="button"> 加载状态
</button>
<script>
   $(function() {
      $(".btn").click(function(){
         $(this).button('loading').delay(1000).queue(function() {
            // $(this).button('reset');
         });
      });
   });
</script>

</body>
</html>
```

结果如下所示：



单个切换

如需激活单个按钮的切换（即改变按钮的正常状态为按压状态，反之亦然），只需向 button 元素添加 **data-toggle="button"** 作为其属性即可，如下面实例所示：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮（Button）插件单个切换</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<button type="button" class="btn btn-primary"
   data-toggle="button"> 单个切换
</button>

</body>
</html>
```

结果如下所示：



复选框（Checkbox）

您可以创建复选框组，并通过向 **btn-group** 添加 data 属性 **data-toggle="buttons"** 来添加复选框组的切换。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮（Button）插件复选框</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group" data-toggle="buttons">
   <label class="btn btn-primary">
      <input type="checkbox"> 选项 1
   </label>
   <label class="btn btn-primary">
      <input type="checkbox"> 选项 2
   </label>
   <label class="btn btn-primary">
      <input type="checkbox"> 选项 3
   </label>
</div>

</body>
</html>
```

结果如下所示：



单选按钮（Radio）

类似地，您可以创建单选按钮组，并通过向 **btn-group** 添加 data 属性 **data-toggle="buttons"** 来添加单选按钮组的切换。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮（Button）插件单选按钮</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="btn-group" data-toggle="buttons">
   <label class="btn btn-primary">
      <input type="radio" name="options" id="option1"> 选项 1
   </label>
   <label class="btn btn-primary">
      <input type="radio" name="options" id="option2"> 选项 2
   </label>
   <label class="btn btn-primary">
      <input type="radio" name="options" id="option3"> 选项 3
   </label>
</div>

</body>
</html>
```

结果如下所示：



用法

您可以 通过 **JavaScript** 启用按钮（Button）插件，如下所示：

```
$('.btn').button()
```

选项

没有选项。

方法

下面是一些按钮（Button）插件中有用的方法：

方法	描述	实例
button('toggle')	切换按压状态。赋予按钮被激活的外观。您可以使用 **data-toggle** 属性启用按钮的自动切换。	`$().button('toggle')`
.button('loading')	当加载时，按钮是禁用的，且文本变为 button 元素的 **data-loading-text** 属性的值。	`$().button('loading')`
.button('reset')	重置按钮状态，文本内容恢复为最初的内容。当您想要把按钮返回为原始的状态时，该方法非常有用。	`$().button('reset')`
.button(string)	该方法中的字符串是指由用户声明的任何字符串。使用该方法，重置按钮状态，并添加新的内容。	`$().button(string)`

实例

下面的实例演示了上面方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 按钮（Button）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<h2>点击每个按钮查看方法效果</h2>
<h4>演示 .button('toggle') 方法</h4>
<div id="myButtons1" class="bs-example">
   <button type="button" class="btn btn-primary">原始</button>
</div>

<h4>演示 .button('loading') 方法</h4>
<div id="myButtons2" class="bs-example">
   <button type="button" class="btn btn-primary"
      data-loading-text="Loading...">原始
   </button>
</div>

<h4>演示 .button('reset') 方法</h4>
```

```
<div id="myButtons3" class="bs-example">
   <button type="button" class="btn btn-primary"
      data-loading-text="Loading...">原始
   </button>
</div>

<h4>演示 .button(string) 方法</h4>
<button type="button" class="btn btn-primary" id="myButton4"
   data-complete-text="Loading finished">请点击我
</button>
<script type="text/javascript">
   $(function () {
      $("#myButtons1 .btn").click(function(){
         $(this).button('toggle');
      });
   });
   $(function() {
      $("#myButtons2 .btn").click(function(){
         $(this).button('loading').delay(1000).queue(function() {
         });
      });
   });
   $(function() {
      $("#myButtons3 .btn").click(function(){
         $(this).button('loading').delay(1000).queue(function() {
            $(this).button('reset');
         });
      });
   });
   $(function() {
      $("#myButton4").click(function(){
         $(this).button('loading').delay(1000).queue(function() {
            $(this).button('complete');
         });
      });
   });
</script>

</body>
</html>
```

结果如下所示：


点击每个按钮查看方法效果
演示 .button('toggle') 方法
原始
演示 .button('loading') 方法
原始
演示 .button('reset') 方法
原始
演示 .button(string) 方法
请点击我

Bootstrap 折叠（Collapse）插件

折叠（Collapse）插件可以很容易地让页面区域折叠起来。无论您用它来创建折叠导航还是内容面板，它都允许很多内容选项。

> 如果您想要单独引用该插件的功能，那么您需要引用 **collapse.js**。同时，也需要在您的 Bootstrap 版本中引用 Transition（过渡）插件。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

您可以使用折叠（Collapse）插件：

- 创建可折叠的分组或折叠面板（*accordion*），如下所示：

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
   &lt;title&gt;Bootstrap 实例 - 折叠面板&lt;/title&gt;
   &lt;link href="/bootstrap/css/bootstrap.min.css" rel="styles
   &lt;script src="/scripts/jquery.min.js"&gt;&lt;/script&gt;
   &lt;script src="/bootstrap/js/bootstrap.min.js"&gt;&lt;/scri
&lt;/head&gt;
&lt;body&gt;

&lt;div class="panel-group" id="accordion"&gt;
  &lt;div class="panel panel-default"&gt;
    &lt;div class="panel-heading"&gt;
      &lt;h4 class="panel-title"&gt;
        &lt;a data-toggle="collapse" data-parent="#accordion"
          href="#collapseOne"&gt;
          点击我进行展开，再次点击我进行折叠。第 1 部分
        &lt;/a&gt;
      &lt;/h4&gt;
    &lt;/div&gt;
    &lt;div id="collapseOne" class="panel-collapse collapse in"
      &lt;div class="panel-body"&gt;
        Nihil anim keffiyeh helvetica, craft beer labore wes ar
        cred nesciunt sapiente ea proident. Ad vegan excepteur
        vice lomo.
      &lt;/div&gt;
    &lt;/div&gt;
  &lt;/div&gt;
  &lt;div class="panel panel-default"&gt;
    &lt;div class="panel-heading"&gt;
      &lt;h4 class="panel-title"&gt;
        &lt;a data-toggle="collapse" data-parent="#accordion"
          href="#collapseTwo"&gt;
          点击我进行展开，再次点击我进行折叠。第 2 部分
        &lt;/a&gt;
```

```
        &lt;/h4&gt;
      &lt;/div&gt;
      &lt;div id="collapseTwo" class="panel-collapse collapse"&gt
        &lt;div class="panel-body"&gt;
          Nihil anim keffiyeh helvetica, craft beer labore wes ar
          cred nesciunt sapiente ea proident. Ad vegan excepteur 
          vice lomo.
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;
    &lt;div class="panel panel-default"&gt;
      &lt;div class="panel-heading"&gt;
        &lt;h4 class="panel-title"&gt;
          &lt;a data-toggle="collapse" data-parent="#accordion"
            href="#collapseThree"&gt;
            点击我进行展开，再次点击我进行折叠。第 3 部分
          &lt;/a&gt;
        &lt;/h4&gt;
      &lt;/div&gt;
      &lt;div id="collapseThree" class="panel-collapse collapse"&
        &lt;div class="panel-body"&gt;
          Nihil anim keffiyeh helvetica, craft beer labore wes ar
          cred nesciunt sapiente ea proident. Ad vegan excepteur 
          vice lomo.
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  &lt;/div&gt;

  &lt;/body&gt;
  &lt;/html&gt;
```

结果如下所示：

点击我进行展开，再次点击我进行折叠。第 1 部分

点击我进行展开，再次点击我进行折叠。第 2 部分

Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo.

点击我进行展开，再次点击我进行折叠。第 3 部分

1. **data-toggle="collapse"** 添加到您想要展开或折叠的组件的链接上。
2. **href** 或 **data-target** 属性添加到父组件，它的值是子组件的 *id*。
3. **data-parent** 属性把折叠面板（accordion）的 id 添加到要展开或折叠的组件的链接上。

- 创建不带 ***accordion*** 标记的简单的可折叠组件（***collapsible***），如下所示：

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
   &lt;title&gt;Bootstrap 实例 - 简单的可折叠组件&lt;/title&gt;
   &lt;link href="/bootstrap/css/bootstrap.min.css" rel="styles
   &lt;script src="/scripts/jquery.min.js"&gt;&lt;/script&gt;
   &lt;script src="/bootstrap/js/bootstrap.min.js"&gt;&lt;/scri
&lt;/head&gt;
&lt;body&gt;

&lt;button type="button" class="btn btn-primary" data-toggle="c
   data-target="#demo"&gt;
   简单的可折叠组件
&lt;/button&gt;

&lt;div id="demo" class="collapse in"&gt;
  Nihil anim keffiyeh helvetica, craft beer labore wes anderson
  cred nesciunt sapiente ea proident. Ad vegan excepteur butche
  vice lomo.
&lt;/div&gt;

&lt;/body&gt;
&lt;/html&gt;
```

结果如下所示：

简单的可折叠组件

Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo.

正如您在实例中看到的，我们创建了一个可折叠的组件，与折叠面板（accordion）不同，我们没有添加属性 **data-parent**。

用法

下表列出了折叠（Collapse）插件用于处理繁重的伸缩的 class：

Class	描述
.collapse	隐藏内容。
.collapse.in	显示内容。
.collapsing	当过渡效果开始时被添加，当过渡效果完成时被移除。

您可以通过以下两种方式使用折叠（Collapse）插件：

- 通过 **data** 属性：向元素添加 **data-toggle="collapse"** 和 **data-target**，自动分配可折叠元素的控制。**data-target** 属性接受一个 CSS 选择器，并会对其应用折叠效果。请确保向可折叠元素添加 class **.collapse**。如果您希望它默认情况下是打开的，请添加额外的 class **.in**。

  为了向可折叠控件添加类似折叠面板的分组管理，请添加 data 属性 **data-parent="#selector"**。

- 通过 **JavaScript**：可通过 JavaScript 激活 collapse 方法，如下所示：

  ```
  $('.collapse').collapse()
  ```

选项

有一些选项是通过 data 属性或 JavaScript 来传递的。下表列出了这些选项：

选项名称	类型/默认值	**Data 属性名称**	描述
parent	selector 默认值：*false*	data-parent	如果提供了一个选择器，当可折叠项目显示时，指定父元素下的所有可折叠的元素将被关闭。这与传统的折叠面板（accordion）的行为类似 - 这依赖于 accordion-group 类。
toggle	boolean 默认值：*true*	data-toggle	切换调用可折叠元素。

方法

下面是一些折叠（Collapse）插件中有用的方法：

方法	描述	实例
Options: .collapse(options)	激活内容为可折叠元素。接受一个可选的 options 对象。	`$('#identifier').collapse({ toggle: fals`
Toggle: .collapse('toggle')	切换显示/隐藏可折叠元素。	`$('#identifier').collapse('toggle')`
Show: .collapse('show')	显示可折叠元素。	`$('#identifier').collapse('show')`
Hide: .collapse('hide')	隐藏可折叠元素。	`$('#identifier').collapse('hide')`

实例

下面的实例演示了方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 折叠（Collapse）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

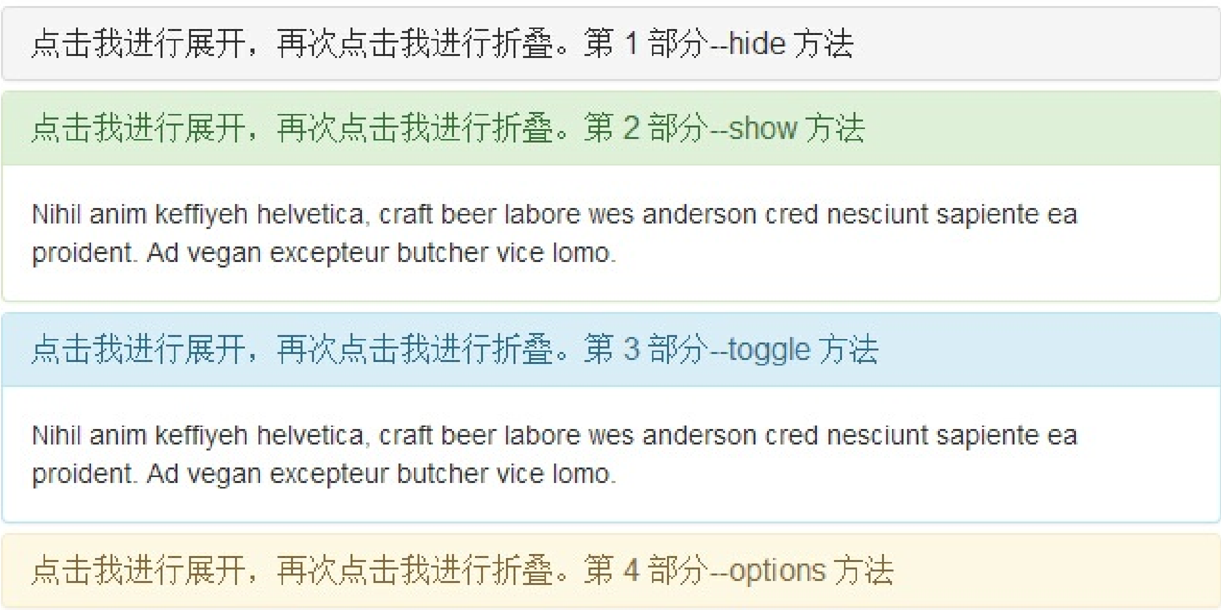
<div class="panel-group" id="accordion">
   <div class="panel panel-default">
      <div class="panel-heading">
         <h4 class="panel-title">
            <a data-toggle="collapse" data-parent="#accordion"
               href="#collapseOne">
               点击我进行展开，再次点击我进行折叠。第 1 部分--hide 方法
            </a>
         </h4>
      </div>
      <div id="collapseOne" class="panel-collapse collapse in">
```

```
            <div class="panel-body">
               Nihil anim keffiyeh helvetica, craft beer labore wes ar
               nesciunt sapiente ea proident. Ad vegan excepteur butch
               lomo.
            </div>
         </div>
      </div>
      <div class="panel panel-success">
         <div class="panel-heading">
            <h4 class="panel-title">
               <a data-toggle="collapse" data-parent="#accordion"
                  href="#collapseTwo">
                  点击我进行展开，再次点击我进行折叠。第 2 部分--show 方法
               </a>
            </h4>
         </div>
         <div id="collapseTwo" class="panel-collapse collapse">
            <div class="panel-body">
               Nihil anim keffiyeh helvetica, craft beer labore wes ar
               nesciunt sapiente ea proident. Ad vegan excepteur butch
               lomo.
            </div>
         </div>
      </div>
      <div class="panel panel-info">
         <div class="panel-heading">
            <h4 class="panel-title">
               <a data-toggle="collapse" data-parent="#accordion"
                  href="#collapseThree">
                  点击我进行展开，再次点击我进行折叠。第 3 部分--toggle 方法
               </a>
            </h4>
         </div>
         <div id="collapseThree" class="panel-collapse collapse">
            <div class="panel-body">
               Nihil anim keffiyeh helvetica, craft beer labore wes ar
               nesciunt sapiente ea proident. Ad vegan excepteur butch
               lomo.
            </div>
         </div>
      </div>
      <div class="panel panel-warning">
         <div class="panel-heading">
            <h4 class="panel-title">
               <a data-toggle="collapse" data-parent="#accordion"
                  href="#collapseFour">
                  点击我进行展开，再次点击我进行折叠。第 4 部分--options 方法
               </a>
            </h4>
         </div>
         <div id="collapseFour" class="panel-collapse collapse">
            <div class="panel-body">
               Nihil anim keffiyeh helvetica, craft beer labore wes ar
```

```
            nesciunt sapiente ea proident. Ad vegan excepteur butch
            lomo.
         </div>
      </div>
   </div>
</div>
<script type="text/javascript">
   $(function () { $('#collapseFour').collapse({
      toggle: false
   })});
   $(function () { $('#collapseTwo').collapse('show')});
   $(function () { $('#collapseThree').collapse('toggle')});
   $(function () { $('#collapseOne').collapse('hide')});
</script>

</body>
</html>
```

结果如下所示：



事件

下表列出了折叠（Collapse）插件中要用到的事件。这些事件可在函数中当钩子使用。

事件	描述	实例
show.bs.collapse	在调用 show 方法后触发该事件。	`$('#identifier').on('show.bs.collapse',`
shown.bs.collapse	当折叠元素对用户可见时触发该事件（将等待 CSS 过渡效果完成）。	`$('#identifier').on('shown.bs.collapse'`
hide.bs.collapse	当调用 hide 实例方法时立即触发该事件。	`$('#identifier').on('hide.bs.collapse',`
hidden.bs.collapse	当折叠元素对用户隐藏时触发该事件（将等待 CSS 过渡效果完成）。	`$('#identifier').on('hidden.bs.collapse`

实例

下面的实例演示了事件的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 折叠（Collapse）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="panel-group" id="accordion">
   <div class="panel panel-info">
      <div class="panel-heading">
         <h4 class="panel-title">
            <a data-toggle="collapse" data-parent="#accordion"
               href="#collapseexample">
               点击我进行展开，再次点击我进行折叠。--shown 事件
            </a>
         </h4>
      </div>
      <div id="collapseexample" class="panel-collapse collapse">
         <div class="panel-body">
            Nihil anim keffiyeh helvetica, craft beer labore wes ar
            cred nesciunt sapiente ea proident.
            Ad vegan excepteur butcher vice lomo.
         </div>
      </div>
   </div>
</div>

<script type="text/javascript">
   $(function () {
      $('#collapseexample').on('show.bs.collapse', function () {
         alert('嘿，当您展开时会提示本警告');})
   });
</script>

</body>
</html>
```

结果如下所示：


点击我进行展开，再次点击我进行折叠。--shown 事件
嘿，当您展开时会提示本警告
确定

Bootstrap 轮播（Carousel）插件

Bootstrap 轮播（Carousel）插件是一种灵活的响应式的向站点添加滑块的方式。除此之外，内容也是足够灵活的，可以是图像、内嵌框架、视频或者其他您想要放置的任何类型的内容。

> 如果您想要单独引用该插件的功能，那么您需要引用 **carousel.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

实例

下面是一个简单的幻灯片，使用 Bootstrap 轮播（Carousel）插件显示了一个循环播放元素的通用组件。为了实现轮播，您只需要添加带有该标记的代码即可。不需要使用 data 属性，只需要简单的基于 class 的开发即可。

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 简单的轮播（Carousel）插件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div id="myCarousel" class="carousel slide">
   <!-- 轮播（Carousel）指标 -->
   <ol class="carousel-indicators">
      <li data-target="#myCarousel" data-slide-to="0" class="active
      <li data-target="#myCarousel" data-slide-to="1"></li>
      <li data-target="#myCarousel" data-slide-to="2"></li>
   </ol>
   <!-- 轮播（Carousel）项目 -->
   <div class="carousel-inner">
      <div class="item active">
         <img src="img/slide1.png" alt="First slide">
      </div>
      <div class="item">
         <img src="img/slide2.png" alt="Second slide">
      </div>
      <div class="item">
         <img src="img/slide3.png" alt="Third slide">
      </div>
   </div>
   <!-- 轮播（Carousel）导航 -->
   <a class="carousel-control left" href="#myCarousel"
      data-slide="prev">&lsaquo;</a>
   <a class="carousel-control right" href="#myCarousel"
      data-slide="next">&rsaquo;</a>
</div>

</body>
</html>
```

结果如下所示：



可选的标题

您可以通过 **.item** 内的 **.carousel-caption** 元素向幻灯片添加标题。只需要在该处放置任何可选的 HTML 即可，它会自动对齐并格式化。下面的实例演示了这点：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 轮播（Carousel）插件的标题</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div id="myCarousel" class="carousel slide">
   <!-- 轮播（Carousel）指标 -->
   <ol class="carousel-indicators">
      <li data-target="#myCarousel" data-slide-to="0" class="active
      <li data-target="#myCarousel" data-slide-to="1"></li>
      <li data-target="#myCarousel" data-slide-to="2"></li>
   </ol>
   <!-- 轮播（Carousel）项目 -->
   <div class="carousel-inner">
      <div class="item active">
         <img src="img/slide1.png" alt="First slide">
         <div class="carousel-caption">标题 1</div>
      </div>
      <div class="item">
         <img src="img/slide2.png" alt="Second slide">
         <div class="carousel-caption">标题 2</div>
      </div>
      <div class="item">
         <img src="img/slide3.png" alt="Third slide">
         <div class="carousel-caption">标题 3</div>
      </div>
   </div>
   <!-- 轮播（Carousel）导航 -->
   <a class="carousel-control left" href="#myCarousel"
      data-slide="prev">&lsaquo;</a>
   <a class="carousel-control right" href="#myCarousel"
      data-slide="next">&rsaquo;</a>
</div>

</body>
</html>
```

结果如下所示：



用法

- 通过 **data** 属性：使用 data 属性可以很容易控制轮播（Carousel）的位置。
  - 属性 **data-slide** 接受关键字 *prev* 或 *next*，用来改变幻灯片相对于当前位置的位置。
  - 使用 **data-slide-to** 来向轮播床底一个原始滑动索引，**data-slide-to="2"** 将把滑块移动到一个特定的索引，索引从 0 开始计数。
  - **data-ride="carousel"** 属性用于标记轮播在页面加载时就开始动画播放。
- 通过 **JavaScript**：轮播（Carousel）可通过 JavaScript 手动调用，如下所示：

```
$('.carousel').carousel()
```

选项

有一些选项是通过 data 属性或 JavaScript 来传递的。下表列出了这些选项：

选项名称	类型/默认值	Data 属性名称	描述
interval	number 默认值：*5000*	data-interval	自动循环每个项目之间延迟的时间量。如果为 false，轮播将不会自动循环。
pause	string 默认值：*"hover"*	data-pause	鼠标进入时暂停轮播循环，鼠标离开时恢复轮播循环。
wrap	boolean 默认值：*true*	data-wrap	轮播是否连续循环。

方法

下面是一些轮播（Carousel）插件中有用的方法：

方法	描述	实例
.carousel(options)	初始化轮播为可选的 options 对象，并开始循环项目。	`$('#identifier').carousel({ interval: 2`
.carousel('cycle')	从左到右循环轮播项目。	`$('#identifier').carousel('cycle')`
.carousel('pause')	停止轮播循环项目。	`$('#identifier')..carousel('pause')`
.carousel(number)	循环轮播到某个特定的帧（从 0 开始计数，与数组类似）。	`$('#identifier').carousel(number)`
.carousel('prev')	循环轮播到上一个项目。	`$('#identifier').carousel('prev')`
.carousel('next')	循环轮播到下一个项目。	`$('#identifier').carousel('next')`

实例

下面的实例演示了方法的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 轮播（Carousel）插件方法</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

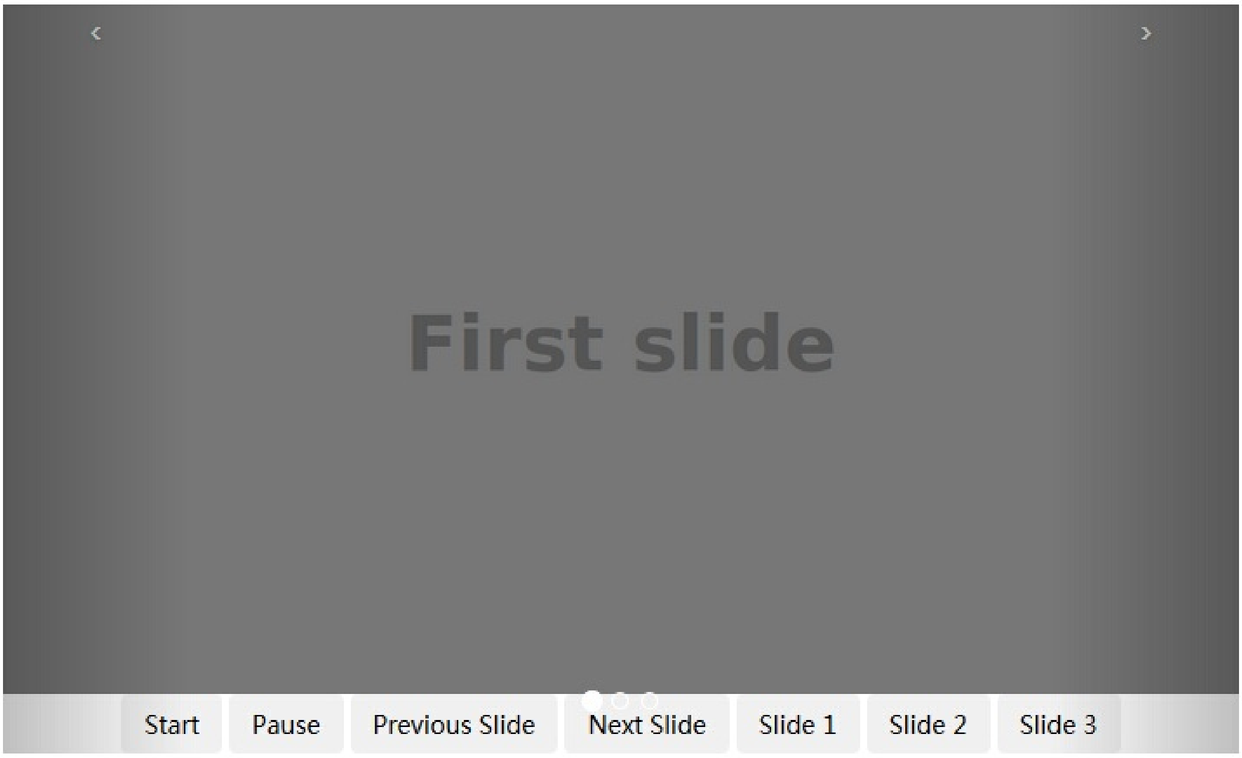
```
    <script src="/scripts/jquery.min.js"></script>
    <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div id="myCarousel" class="carousel slide">
   <!-- 轮播（Carousel）指标 -->
   <ol class="carousel-indicators">
      <li data-target="#myCarousel" data-slide-to="0"
         class="active"></li>
      <li data-target="#myCarousel" data-slide-to="1"></li>
      <li data-target="#myCarousel" data-slide-to="2"></li>
   </ol>
   <!-- 轮播（Carousel）项目 -->
   <div class="carousel-inner">
      <div class="item active">
         <img src="img/slide1.png" alt="First slide">
      </div>
      <div class="item">
         <img src="img/slide2.png" alt="Second slide">
      </div>
      <div class="item">
         <img src="img/slide3.png" alt="Third slide">
      </div>
   </div>
   <!-- 轮播（Carousel）导航 -->
   <a class="carousel-control left" href="#myCarousel"
      data-slide="prev">&lsaquo;</a>
   <a class="carousel-control right" href="#myCarousel"
      data-slide="next">&rsaquo;</a>
   <!-- 控制按钮 -->
   <div style="text-align:center;">
      <input type="button" class="btn start-slide" value="Start">
      <input type="button" class="btn pause-slide" value="Pause">
      <input type="button" class="btn prev-slide" value="Previous S
      <input type="button" class="btn next-slide" value="Next Slide
      <input type="button" class="btn slide-one" value="Slide 1">
      <input type="button" class="btn slide-two" value="Slide 2">
      <input type="button" class="btn slide-three" value="Slide 3">
   </div>
</div>
<script>
   $(function(){
      // 初始化轮播
      $(".start-slide").click(function(){
         $("#myCarousel").carousel('cycle');
      });
      // 停止轮播
      $(".pause-slide").click(function(){
         $("#myCarousel").carousel('pause');
      });
      // 循环轮播到上一个项目
      $(".prev-slide").click(function(){
```

```
            $("#myCarousel").carousel('prev');
        });
        // 循环轮播到下一个项目
        $(".next-slide").click(function(){
            $("#myCarousel").carousel('next');
        });
        // 循环轮播到某个特定的帧
        $(".slide-one").click(function(){
            $("#myCarousel").carousel(0);
        });
        $(".slide-two").click(function(){
            $("#myCarousel").carousel(1);
        });
        $(".slide-three").click(function(){
            $("#myCarousel").carousel(2);
        });
    });
</script>

</body>
</html>
```

结果如下所示：



事件

下表列出了轮播（Carousel）插件中要用到的事件。这些事件可在函数中当钩子使用。

事件	描述	实例
slide.bs.carousel	当调用 slide 实例方法时立即触发该事件。	`$('#identifier').on('slide.bs.carousel', fu`
slid.bs.carousel	当轮播完成幻灯片过渡效果时触发该事件。	`$('#identifier').on('slid.bs.carousel', fun`

实例

下面的实例演示了事件的用法：

```
<!DOCTYPE html>
<html>
<head>
   <title>Bootstrap 实例 - 轮播（Carousel）插件事件</title>
   <link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">
   <script src="/scripts/jquery.min.js"></script>
   <script src="/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div id="myCarousel" class="carousel slide">
   <!-- 轮播（Carousel）指标 -->
   <ol class="carousel-indicators">
      <li data-target="#myCarousel" data-slide-to="0"
         class="active"></li>
      <li data-target="#myCarousel" data-slide-to="1"></li>
      <li data-target="#myCarousel" data-slide-to="2"></li>
   </ol>
   <!-- 轮播（Carousel）项目 -->
   <div class="carousel-inner">
      <div class="item active">
         <img src="img/slide1.png" alt="First slide">
      </div>
      <div class="item">
         <img src="img/slide2.png" alt="Second slide">
      </div>
      <div class="item">
         <img src="img/slide3.png" alt="Third slide">
      </div>
   </div>
   <!-- 轮播（Carousel）导航 -->
   <a class="carousel-control left" href="#myCarousel"
      data-slide="prev">&lsaquo;</a>
   <a class="carousel-control right" href="#myCarousel"
      data-slide="next">&rsaquo;</a>
</div>
<script>
   $(function(){
      $('#myCarousel').on('slide.bs.carousel', function () {
         alert("当调用 slide 实例方法时立即触发该事件。");
      });
   });
</script>

</body>
</html>
```

结果如下所示：


当调用 slide 实例方法时立即触发该事件。
确定

Bootstrap 附加导航（Affix）插件

附加导航（Affix）插件允许某个 <div> 固定在页面的某个位置。您也可以在打开或关闭使用该插件之间进行切换。一个常见的例子是社交图标。它们将在某个位置开始，但当页面点击某个标记，该 <div> 会锁定在某个位置，不会随着页面其他部分一起滚动。

> 如果您想要单独引用该插件的功能，那么您需要引用 **affix.js**。或者，正如 Bootstrap 插件概览 一章中所提到，您可以引用 *bootstrap.js* 或压缩版的 *bootstrap.min.js*。

用法

您可以通过 data 属性或者通过 JavaScript 来使用附加导航（Affix）插件。

- 通过 **data** 属性：如需向元素添加附加导航（Affix）行为，只需要向需要监听的元素添加 **data-spy="affix"** 即可。请使用偏移来定义何时切换元素的锁定和移动。

实例

下面的实例演示了通过 data 属性使用附加导航（Affix）插件的用法：

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta charset="UTF-8"&gt;
&lt;title&gt;Bootstrap 附加导航（Affix）插件&lt;/title&gt;
&lt;link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet"&
&lt;script src="/scripts/jquery.min.js"&gt;&lt;/script&gt;
&lt;script src="/bootstrap/js/bootstrap.min.js"&gt;&lt;/script&gt;

&lt;style type="text/css"&gt;
    /* Custom Styles */
    ul.nav-tabs{
        width: 140px;
        margin-top: 20px;
        border-radius: 4px;
        border: 1px solid #ddd;
        box-shadow: 0 1px 4px rgba(0, 0, 0, 0.067);
    }
    ul.nav-tabs li{
        margin: 0;
        border-top: 1px solid #ddd;
    }
    ul.nav-tabs li:first-child{
```

```
        border-top: none;
    }
    ul.nav-tabs li a{
        margin: 0;
        padding: 8px 16px;
        border-radius: 0;
    }
    ul.nav-tabs li.active a, ul.nav-tabs li.active a:hover{
        color: #fff;
        background: #0088cc;
        border: 1px solid #0088cc;
    }
    ul.nav-tabs li:first-child a{
        border-radius: 4px 4px 0 0;
    }
    ul.nav-tabs li:last-child a{
        border-radius: 0 0 4px 4px;
    }
    ul.nav-tabs.affix{
        top: 30px; /* Set the top position of pinned element */
    }
&lt;/style&gt;
&lt;/head&gt;
&lt;body data-spy="scroll" data-target="#myScrollspy"&gt;
&lt;div class="container"&gt;
    &lt;div class="jumbotron"&gt;
        &lt;h1&gt;Bootstrap Affix&lt;/h1&gt;
    &lt;/div&gt;
    &lt;div class="row"&gt;
        &lt;div class="col-xs-3" id="myScrollspy"&gt;
            &lt;ul class="nav nav-tabs nav-stacked" data-spy="affix
                &lt;li class="active"&gt;&lt;a href="#section-1"&gt
                &lt;li&gt;&lt;a href="#section-2"&gt;第二部分&lt;/a&
                &lt;li&gt;&lt;a href="#section-3"&gt;第三部分&lt;/a&
                &lt;li&gt;&lt;a href="#section-4"&gt;第四部分&lt;/a&
                &lt;li&gt;&lt;a href="#section-5"&gt;第五部分&lt;/a&
            &lt;/ul&gt;
        &lt;/div&gt;
        &lt;div class="col-xs-9"&gt;
            &lt;h2 id="section-1"&gt;第一部分&lt;/h2&gt;
            &lt;p&gt;Lorem ipsum dolor sit amet, consectetur adipis
            &lt;p&gt;Vestibulum quis quam ut magna consequat faucib
            &lt;hr&gt;
            &lt;h2 id="section-2"&gt;第二部分&lt;/h2&gt;
            &lt;p&gt;Nullam hendrerit justo non leo aliquet imperdi
            &lt;p&gt;Vestibulum consectetur scelerisque lacus, ac f
            &lt;hr&gt;
            &lt;h2 id="section-3"&gt;第三部分&lt;/h2&gt;
            &lt;p&gt;Integer pulvinar leo id risus pellentesque ves
            &lt;p&gt;Phasellus vitae suscipit justo. Mauris pharetr
            &lt;p&gt;Quisque pharetra velit id velit iaculis pretiu
            &lt;hr&gt;
            &lt;h2 id="section-4"&gt;第四部分&lt;/h2&gt;
```

```
                &lt;p&gt;Suspendisse a orci facilisis, dignissim tortor
                &lt;p&gt;Vestibulum quis quam ut magna consequat faucib
                &lt;p&gt;Phasellus fermentum, neque sit amet sodales te
                &lt;p&gt;Ut ut risus nisl. Fusce porttitor eros at magr
                &lt;hr&gt;
                &lt;h2 id="section-5"&gt;第五部分&lt;/h2&gt;
                &lt;p&gt;Nam eget purus nec est consectetur vehicula. N
                &lt;p&gt;Vivamus mattis accumsan erat, vel convallis ri
                &lt;p&gt;Suspendisse a orci facilisis, dignissim tortor
                &lt;p&gt;Morbi sed fermentum ipsum. Morbi a orci vulput
                &lt;p&gt;Sed vitae lobortis diam, id molestie magna. Al
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
    &lt;/body&gt;
    &lt;/html&gt;
```

结果如下所示：



- 通过 **JavaScript**：您可以通过 JavaScript 手动为某个元素添加附加导航（Affix），如下所示：

```
$('#myAffix').affix({
   offset: {
      top: 100, bottom: function () {
         return (this.bottom =
            $('.bs-footer').outerHeight(true))
         }
      }
})
```

实例

下面的实例演示了通过 JavaScript 使用附加导航（Affix）插件的用法：

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta charset="UTF-8"&gt;
&lt;title&gt;Bootstrap 附加导航（Affix）插件&lt;/title&gt;
&lt;link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet"&g
&lt;script src="/scripts/jquery.min.js"&gt;&lt;/script&gt;
&lt;script src="/bootstrap/js/bootstrap.min.js"&gt;&lt;/script&gt;
&lt;style type="text/css"&gt;
    /* Custom Styles */
    ul.nav-tabs{
        width: 140px;
        margin-top: 20px;
        border-radius: 4px;
        border: 1px solid #ddd;
        box-shadow: 0 1px 4px rgba(0, 0, 0, 0.067);
    }
    ul.nav-tabs li{
        margin: 0;
        border-top: 1px solid #ddd;
    }
    ul.nav-tabs li:first-child{
        border-top: none;
    }
    ul.nav-tabs li a{
        margin: 0;
        padding: 8px 16px;
        border-radius: 0;
    }
    ul.nav-tabs li.active a, ul.nav-tabs li.active a:hover{
        color: #fff;
        background: #0088cc;
        border: 1px solid #0088cc;
    }
    ul.nav-tabs li:first-child a{
        border-radius: 4px 4px 0 0;
    }
```

```
    ul.nav-tabs li:last-child a{
        border-radius: 0 0 4px 4px;
    }
    ul.nav-tabs.affix{
        top: 30px; /* Set the top position of pinned element */
    }
&lt;/style&gt;
&lt;script type="text/javascript"&gt;
$(document).ready(function(){
    $("#myNav").affix({
        offset: {
            top: 125
         }
    });
});
&lt;/script&gt;
&lt;/head&gt;
&lt;body data-spy="scroll" data-target="#myScrollspy"&gt;
&lt;div class="container"&gt;
    &lt;div class="jumbotron"&gt;
        &lt;h1&gt;Bootstrap Affix&lt;/h1&gt;
    &lt;/div&gt;
    &lt;div class="row"&gt;
        &lt;div class="col-xs-3" id="myScrollspy"&gt;
            &lt;ul class="nav nav-tabs nav-stacked" id="myNav"&gt;
                &lt;li class="active"&gt;&lt;a href="#section-1"&gt
                &lt;li&gt;&lt;a href="#section-2"&gt;第二部分&lt;/a&
                &lt;li&gt;&lt;a href="#section-3"&gt;第三部分&lt;/a&
                &lt;li&gt;&lt;a href="#section-4"&gt;第四部分&lt;/a&
                &lt;li&gt;&lt;a href="#section-5"&gt;第五部分&lt;/a&
            &lt;/ul&gt;
        &lt;/div&gt;
        &lt;div class="col-xs-9"&gt;
            &lt;h2 id="section-1"&gt;第一部分&lt;/h2&gt;
            &lt;p&gt;Lorem ipsum dolor sit amet, consectetur adipis
            &lt;p&gt;Vestibulum quis quam ut magna consequat faucib
            &lt;hr&gt;
            &lt;h2 id="section-2"&gt;第二部分&lt;/h2&gt;
            &lt;p&gt;Nullam hendrerit justo non leo aliquet imperdi
            &lt;p&gt;Vestibulum consectetur scelerisque lacus, ac f
            &lt;hr&gt;
            &lt;h2 id="section-3"&gt;第三部分&lt;/h2&gt;
            &lt;p&gt;Integer pulvinar leo id risus pellentesque ves
            &lt;p&gt;Phasellus vitae suscipit justo. Mauris pharetr
            &lt;p&gt;Quisque pharetra velit id velit iaculis pretiu
            &lt;hr&gt;
            &lt;h2 id="section-4"&gt;第四部分&lt;/h2&gt;
            &lt;p&gt;Suspendisse a orci facilisis, dignissim tortor
            &lt;p&gt;Vestibulum quis quam ut magna consequat faucib
            &lt;p&gt;Phasellus fermentum, neque sit amet sodales te
            &lt;p&gt;Ut ut risus nisl. Fusce porttitor eros at magn
            &lt;hr&gt;
            &lt;h2 id="section-5"&gt;第五部分&lt;/h2&gt;
```

```
                &lt;p&gt;Nam eget purus nec est consectetur vehicula. N
                &lt;p&gt;Vivamus mattis accumsan erat, vel convallis ri
                &lt;p&gt;Suspendisse a orci facilisis, dignissim tortor
                &lt;p&gt;Morbi sed fermentum ipsum. Morbi a orci vulput
                &lt;p&gt;Sed vitae lobortis diam, id molestie magna. Al
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
    &lt;/body&gt;
    &lt;/html&gt;
```

结果如下所示：



通过 CSS 定位

在上面两种使用附加导航（Affix）插件的方式中，您都必须通过 CSS 定位内容。附加导航（Affix）插件在三种 class 之间切换，每种 class 都呈现了特定的状态：*.affix*、*.affix-top* 和 *.affix-bottom*。请按照下面的步骤，来为这三种状态设置您自己的 CSS（不依赖此插件）。

- 在开始时，插件添加 **.affix-top** 来指示元素在它的最顶端位置。这个时候不需要任何的 CSS 定位。
- 当滚动经过添加了附加导航（Affix）的元素时，应触发实际的附加导航（Affix）。此时 **.affix** 会替代 **.affix-top**，同时设置 **position: fixed;**（由 Bootstrap 的 CSS 代码提供）。
- 如果定义了底部偏移，当滚动到达该位置时，应把 **.affix** 替换为 **.affix-bottom**。由于偏移是可选的，假如设置了该偏移，则要求同时设置适当的 CSS。在这种情况下，请在必要的时候添加 **position: absolute;**。

选项

有一些选项是通过 data 属性或 JavaScript 来传递的。下表列出了这些选项：

选项名称	类型/默认值	**Data** 属性名称	描述
offset	number | function | object 默认值：*10*	data-offset	当计算滚动位置时，距离顶部的偏移像素。如果设置了一个数字，则该偏移量的值将被应用在顶部和底部。如果设置了一个对象偏移，则其值形如 offset: { top: 10 } 或 offset: { top: 10, bottom: 5 }。如果需要动态计算偏移，请使用函数。

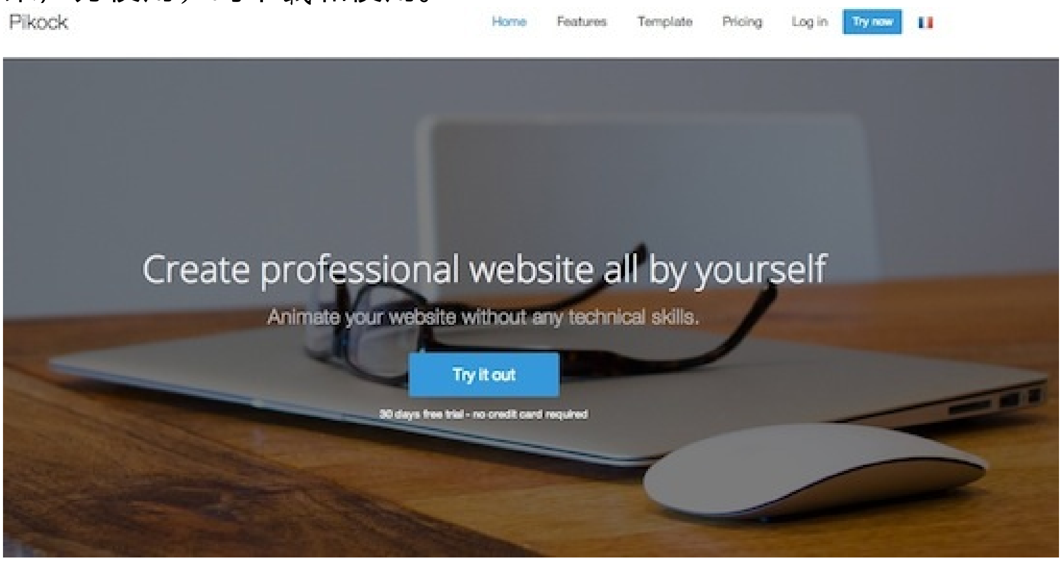
Bootstrap 其它

Bootstrap UI 编辑器

以下是 15 款最好的 Bootstrap 编辑器或者是在线编辑工具。

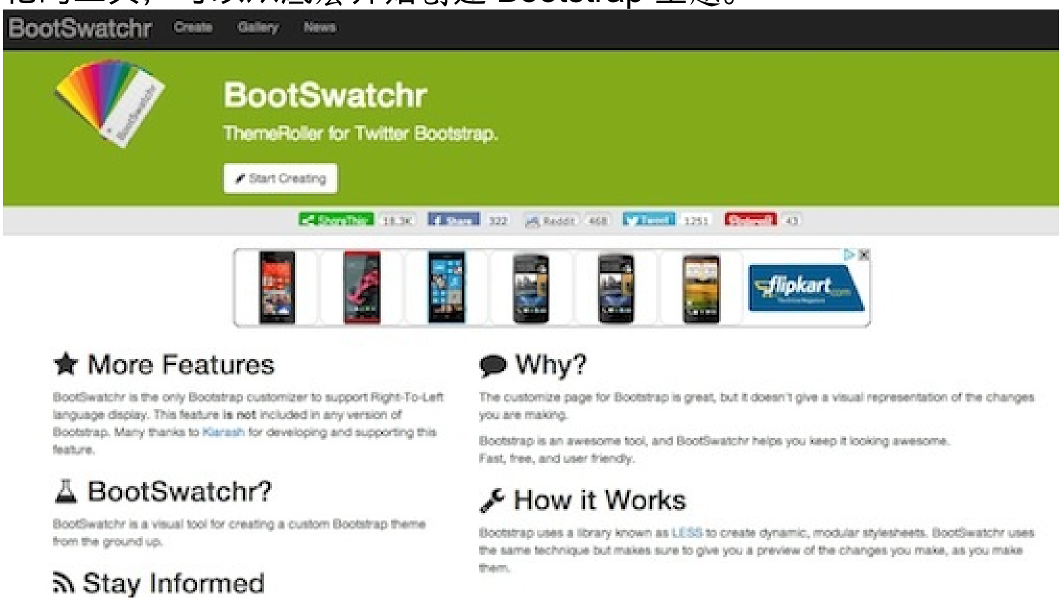
1. Bootstrap Magic

这是一个 Bootstrap 主题生成器，使用最新的 Bootstrap 3 版本和 Angular JS 版本，提供一个鲜活的用户修改预览。它包括了各种各样的导入，一个颜色选择器和智能的预先输入。更神奇的是，Bootstrap 会根据用户的每个选择来重新简历框架，方便用户的下载和使用。



2. BootSwatchr

BootSwatchr 是由 Drew Strickiand 独立开发和维护的，是唯一支持从右到左语言显示的 Bootstrap 自定义构建工具，这也是它的特色之一。BootSwatchr 是个可视化的工具，可以从底层开始创建 Bootstrap 主题。



3. Bootstrap Live Editor

Bootstrap Live Editor 是个所见即所得的构建 Bootstrap 的编辑器。在 Bootstrap 准备好的内容和代码片段的基础上提供一个又好又优雅的方法来编辑和美化 html，也因此它是个明智的选择。除此之外，它还包括高级的选项比如按钮配置，自定义标签和为用户做最明智最有益的选择。



4. Fancy Boot

Fancyboot 是个非常有用的 Bootstrap 自定义工具，允许用户编辑 Bootstrap 的配置，自动预览他们的修改结果。通过滑动菜单可以选择需要使用的插件和组件。当用户做好了相应的修改，可以下载一个包含用户所使用的 jQuery 插件的编译好的和简化后的 CSS 文件。

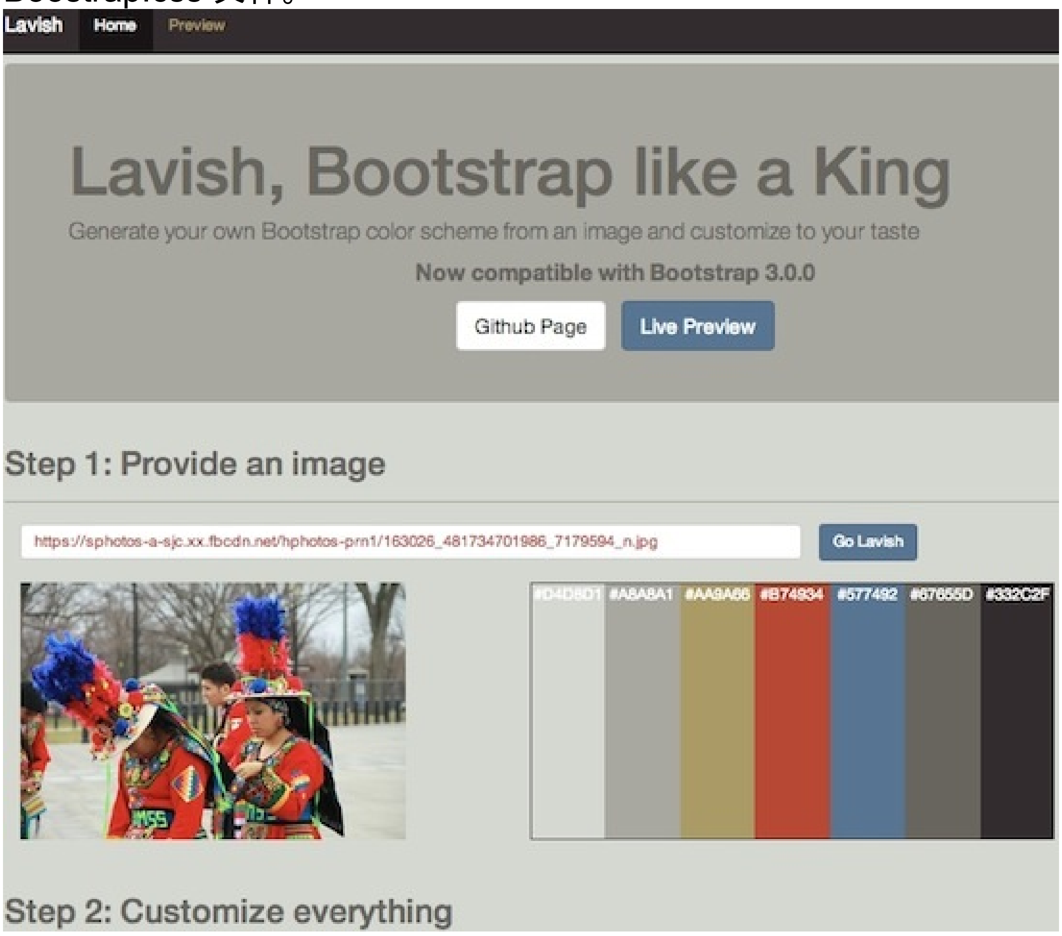


5. Style Bootstrap



6. Lavish

Lavish 是个帮助用户从一个图像中定制自己喜欢的 Bootstrap 配色方案。用户可以从图像中选择各种颜色用在 body 背景，链接，头部，菜单，标签，文本，下拉菜单等等元素上。一旦你确定了 Bootstrap 配色方案，就可以很方便的下载相应的 Boostrap.css 文件。



7. Bootstrap ThemeRoller

7. Bootstrap ThemeRoller

Bootstrap ThemeRoller 是个直观的 web 应用程序， 提供一个可视化的界面让用户按照自己的风格创建属于自己的 Bootstrap 主题。有了这个工具，用户可以自定义颜色，大小，字体，格式，表，按钮，警告信息，导航条，菜单等等元素。 这个应用还会帮用户创建的每种风格都生成唯一的一个 URL， 这样就可以保存用户所有的设置。更不可思议的是，当用户离开的时候，用户可以停止或者恢复自己定制的



页面。

8. LayoutIt!

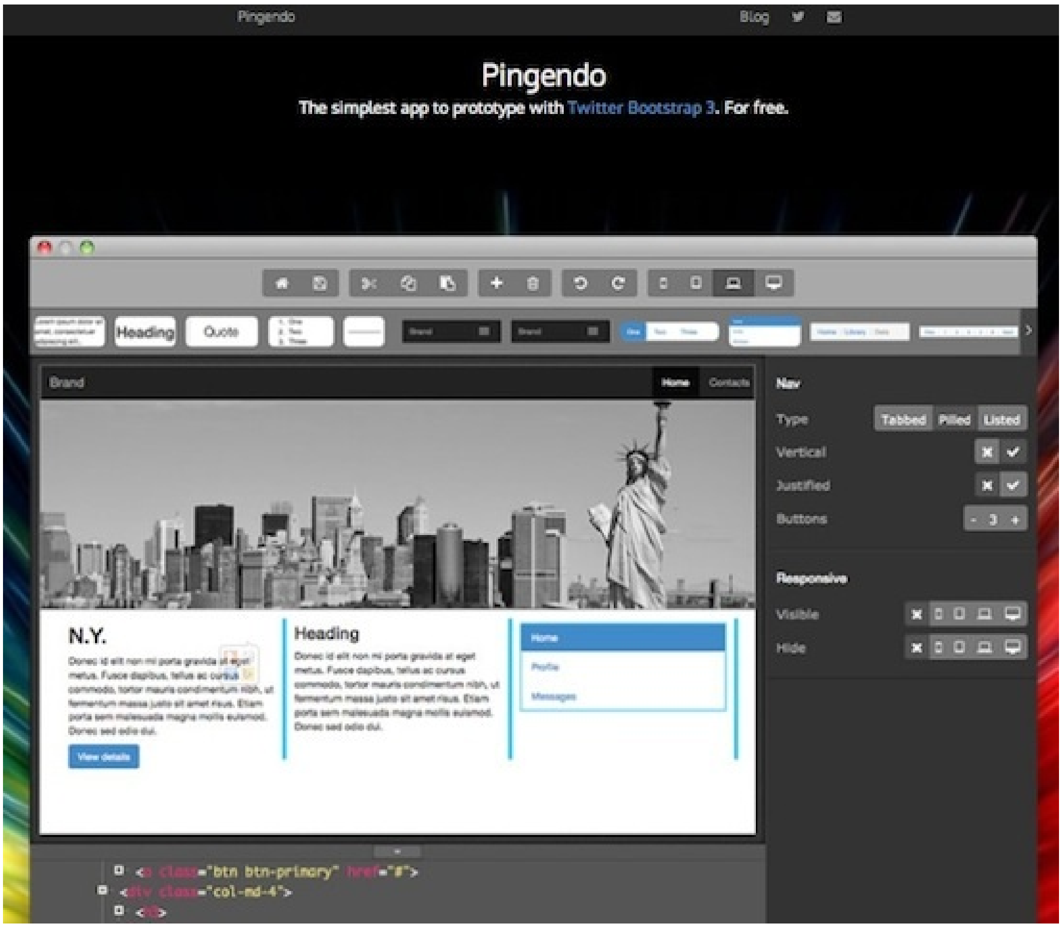
LayoutIt！拥有拖拽接口的功能，能简单迅速的构建一个 Bootstrap 前端代码。LayoutIt！ 兼容任何的编程语言，允许用户下载 HTML， 在这里自由的进行编码设



计。

9. Pingendo

Pingendo 是个可视化的桌面应用程序，允许用户给予流行的 Twitter Bootstrap 框架创建响应式 web 页面原型。Pingendo 还提供大量的现成的布局，拖拽元素的位置，调整大小和自定义的功能。用户可以自行优化他们的页面，插入他们自己的内

容。

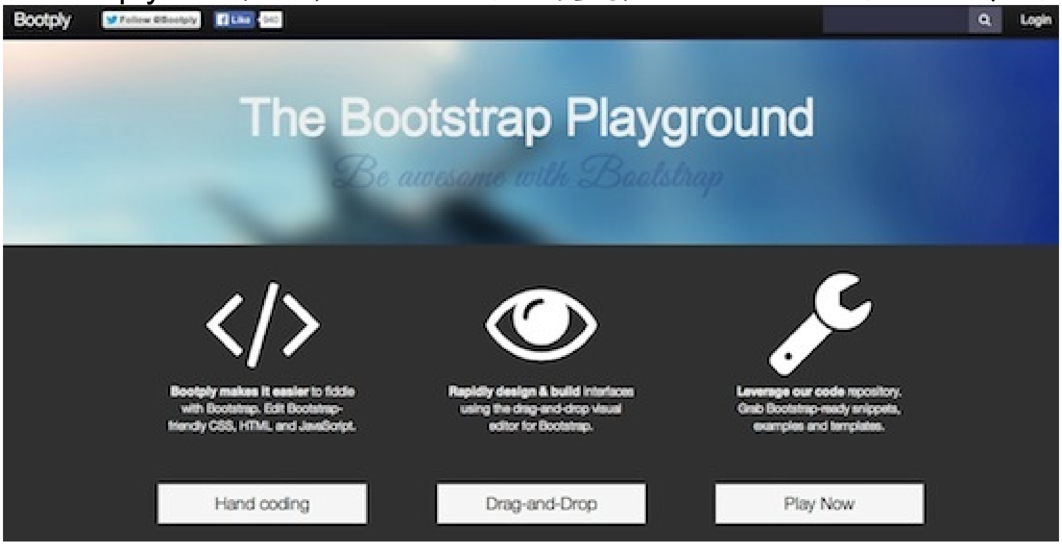
10. Kickstrap

Kickstrap 是个 Bootstrap UI 编辑器，使用 Firebase 作为一个后台服务。它无缝的接合了 Bootstrap 和顶级的 web 技术。更厉害的是，它可以运行一个已验证的，不需要本地后台支持，数据库驱动的 web 应用。



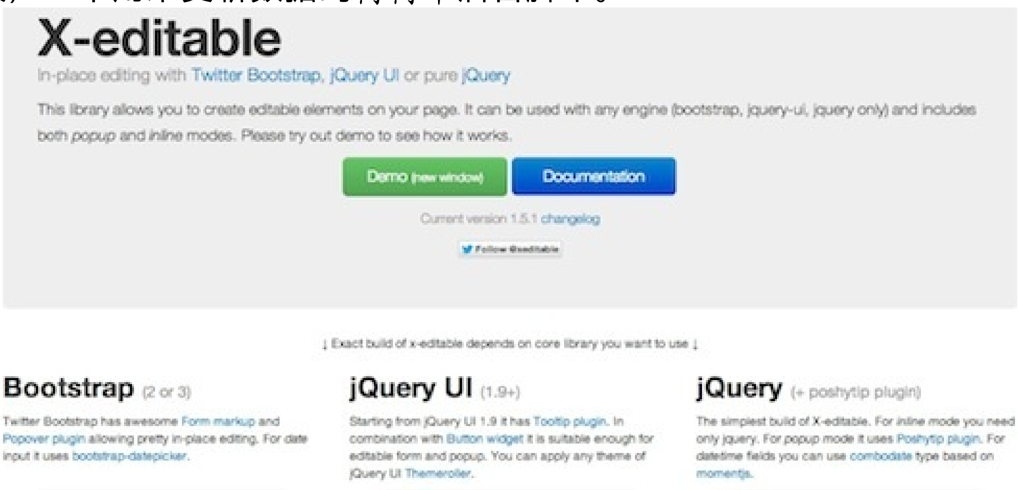
11. Bootply

Bootply 是一款很棒的 Bootstrap UI 编辑工具，尽可能简单的让用户编辑 Bootstrap 友好的 CSS，HTML 和 Javascript。同时也是一个可拖拽的可视化编辑器，用来设计，创建原型和测试 Bootstrap 框架。Bootply 有个代码存储库，里面包括了 Bootstrap 代码片段，示例和模板。除了以上这些，更吸引人的是它有个非常活跃的 Bootply 社区，用户可以很方便的发现，分享和展示相关的代码和片段。



12. X-editable

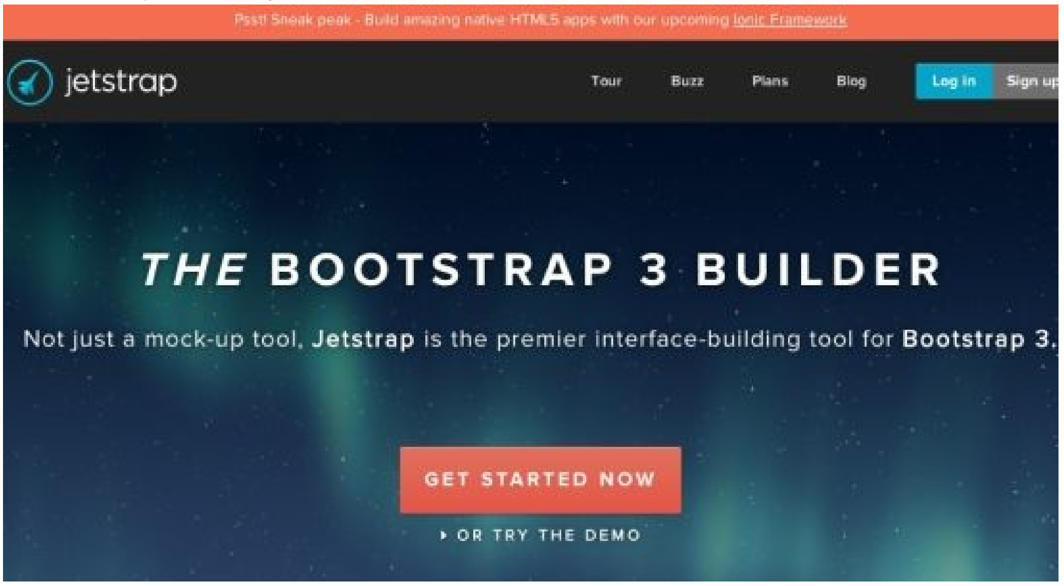
X-Editable 是一个扩展库，允许用户创建可编辑的元素，比如页面弹出和内联模式。她提供本地的 Bootstrap，jQuery UI 或者是纯 jQuery 编辑功能。此 Bootstrap UI 编辑器拥有许多很强大的功能，包括客户端，服务端验证，支持键盘，live 事件，可定制的容器空间等等。总而言之，它提供非常快速的开发功能，易于删除字段，一个用来更新数据的简简单后台脚本。



13. Jetstrap

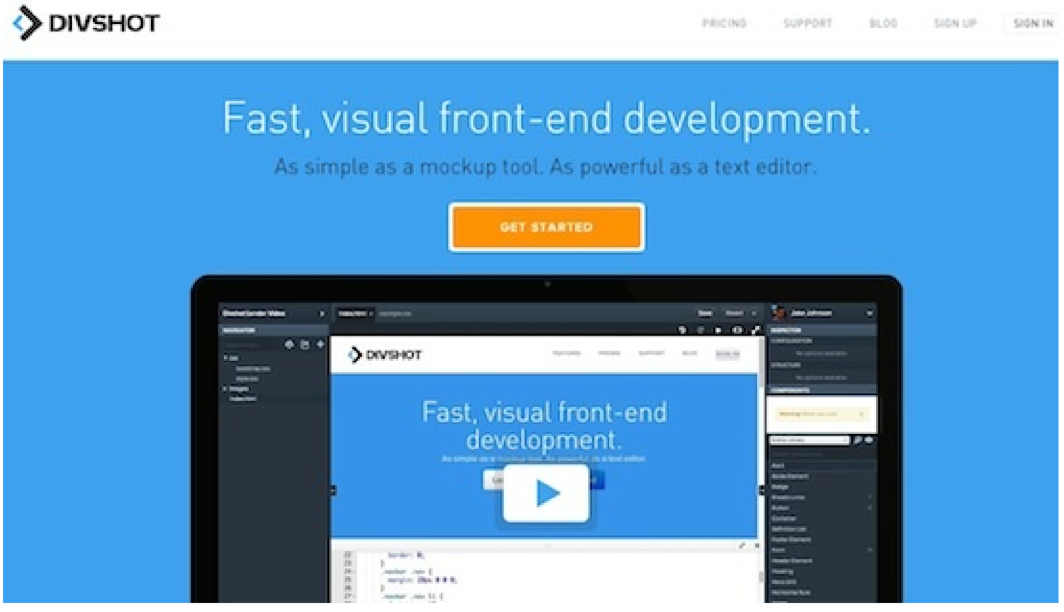
不要错误的以为 Jetstrap 只是个简单的实体模型工具，它是 100% 基于 web 接口的 Twitter Bootstrap 构建工具。用户不需要下载任何类型的软件就可以使用 Jetstrap。用户只需要登录，然后构建自己的项目，就可以随时随地进行访问了。

构建功能主要是提供给开发者和设计者的，用户很容易就可以通过 Jetstrap 生成漂亮的网站，而且运行速度非常快



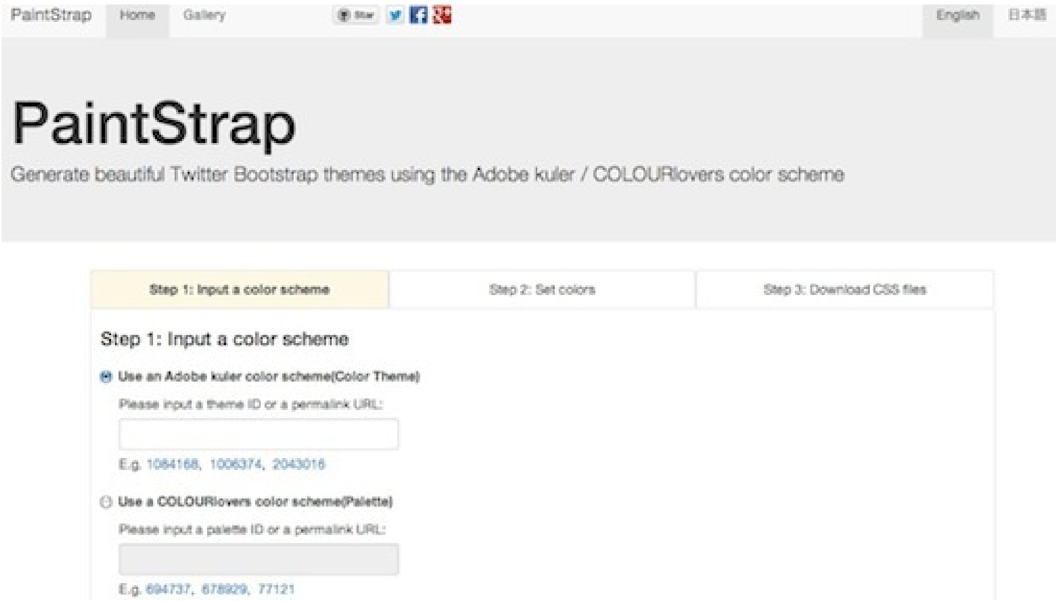
14. DivShot

DivShot 是个快速的，可视化的前端开发文本编辑器，跟 mockup 工具一样的简单整洁。包括各种框架需要的各种定制的组件库，这是它提供的最好的工具之一。除此之外还包括了一个主题转换器，允许用户修改主题。试验版本是免费的，过了试验期就需要自己付费维持原来的服务了。



15. PaintStrap

PaintStrap 使用 Adobe kuler/COLOURlovers 颜色组合来生成漂亮的 Twitter Bootstrap 主题。这个工具特别容易安装，同时很容易下载 CSS 文件并且也很容易替换成另一个标准的文件。


PaintStrap
Home
Gallery
English
日本語
PaintStrap
Generate beautiful Twitter Bootstrap themes using the Adobe kuler / COLOURlovers color scheme
Step 1: Input a color scheme
Step 2: Set colors
Step 3: Download CSS files
Step 1: Input a color scheme
Use an Adobe kuler color scheme(Color Theme)
Please input a theme ID or a permalink URL:
E.g. 1084168, 1006374, 2043016
Use a COLOURlovers color scheme(Palette)
Please input a palette ID or a permalink URL:
E.g. 694737, 678929, 77121

Bootstrap HTML编码规范

语法

- 用两个空格来代替制表符（tab） -- 这是唯一能保证在所有环境下获得一致展现的方法。
- 嵌套元素应当缩进一次（即两个空格）。
- 对于属性的定义，确保全部使用双引号，绝不要使用单引号。
- 不要在自闭合（self-closing）元素的尾部添加斜线 -- HTML5 规范中明确说明这是可选的。
- 不要省略可选的结束标签（closing tag）（例如，`</li>` 或 `</body>` ）。

实例：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <img src="images/company-logo.png" alt="Company">
    <h1 class="hello-world">Hello, world!</h1>
  </body>
</html>
```

HTML5 doctype

为每个 HTML 页面的第一行添加标准模式（standard mode）的声明，这样能够确保在每个浏览器中拥有一致的展现。

实例：

```
<!DOCTYPE html>
<html>
  <head>
  </head>
</html>
```

语言属性

根据 HTML5 规范：

强烈建议为 html 根元素指定 lang 属性，从而为文档设置正确的语言。这将有助于语音合成工具确定其所应该采用的发音，有助于翻译工具确定其翻译时所应遵守的规则等等。

更多关于 `lang` 属性的知识可以从 此规范 中了解。

这里列出了语言代码表。

```
<html lang="zh-CN">
  <!-- ... -->
</html>
```

IE 兼容模式

IE 支持通过特定的 `<meta>` 标签来确定绘制当前页面所应该采用的 IE 版本。除非有强烈的特殊需求，否则最好是设置为 **edge mode**，从而通知 IE 采用其所支持的最新的模式。

```
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
```

字符编码

通过明确声明字符编码，能够确保浏览器快速并容易的判断页面内容的渲染方式。这样做的好处是，可以避免在 HTML 中使用字符实体标记（character entity），从而全部与文档编码一致（一般采用 UTF-8 编码）。

```
<head>
  <meta charset="UTF-8">
</head>
```

引入 CSS 和 JavaScript 文件

根据 HTML5 规范，在引入 CSS 和 JavaScript 文件时一般不需要指定 `type` 属性，因为 `text/css` 和 `text/javascript` 分别是它们的默认值。

HTML5 spec links

- Using link
- Using style
- Using script

```
<!-- External CSS -->
<link rel="stylesheet" href="code-guide.css">

<!-- In-document CSS -->
<style>
  /* ... */
</style>

<!-- JavaScript -->
<script src="code-guide.js"></script>
```

实用为王

尽量遵循 HTML 标准和语义，但是不要以牺牲实用性为代价。任何时候都要尽量使用最少的标签并保持最小的复杂度。

属性顺序

HTML 属性应当按照以下给出的顺序依次排列，确保代码的易读性。

- `class`
- `id` , `name`
- `data-*`
- `src` , `for` , `type` , `href`
- `title` , `alt`
- `aria-*` , `role`

class 用于标识高度可复用组件，因此应该排在首位。id 用于标识具体组件，应当谨慎使用（例如，页面内的书签），因此排在第二位。

```
<a class="..." id="..." data-modal="toggle" href="#">
  Example link
</a>

<input class="form-control" type="text">

<img src="..." alt="...">
```

布尔（boolean）型属性

布尔型属性可以在声明时不赋值。XHTML 规范要求为其赋值，但是 HTML5 规范不需要。

更多信息请参考 WhatWG section on boolean attributes ：

元素的布尔型属性如果有值，就是 true，如果没有值，就是 false。

如果一定要为其赋值的话，请参考 WhatWG 规范：

如果属性存在，其值必须是空字符串或 [...] 属性的规范名称，并且不要再收尾添加空白符。

简单来说，就是不用赋值。

```
<input type="text" disabled>

<input type="checkbox" value="1" checked>

<select>
  <option value="1" selected>1</option>
</select>
```

减少标签的数量

编写 HTML 代码时，尽量避免多余的父元素。很多时候，这需要迭代和重构来实现。请看下面的案例：

```
<!-- Not so great -->
<span class="avatar">
  <img src="...">
</span>

<!-- Better -->
<img class="avatar" src="...">
```

JavaScript 生成的标签

通过 JavaScript 生成的标签让内容变得不易查找、编辑，并且降低性能。能避免时尽量避免。

Bootstrap CSS编码规范

语法

- 用两个空格来代替制表符（tab） -- 这是唯一能保证在所有环境下获得一致展现的方法。
- 为选择器分组时，将单独的选择器单独放在一行。
- 为了代码的易读性，在每个声明块的左花括号前添加一个空格。
- 声明块的右花括号应当单独成行。
- 每条声明语句的 `:` 后应该插入一个空格。
- 为了获得更准确的错误报告，每条声明都应该独占一行。
- 所有声明语句都应当以分号结尾。最后一条声明语句后面的分号是可选的，但是，如果省略这个分号，你的代码可能更易出错。
- 对于以逗号分隔的属性值，每个逗号后面都应该插入一个空格（例如， `box-shadow` ）。
- 不要在 `rgb()` 、 `rgba()` 、 `hsl()` 、 `hsla()` 或 `rect()` 值的内部的逗号后面插入空格。这样利于从多个属性值（既加逗号也加空格）中区分多个颜色值（只加逗号，不加空格）。
- 对于属性值或颜色参数，省略小于 1 的小数前面的 0 （例如， `.5` 代替 `0.5` ； `-.5px` 代替 `-0.5px` ）。
- 十六进制值应该全部小写，例如， `#fff` 。在扫描文档时，小写字符易于分辨，因为他们的形式更易于区分。
- 尽量使用简写形式的十六进制值，例如，用 `#fff` 代替 `#ffffff` 。
- 为选择器中的属性添加双引号，例如， `input[type="text"]` 。只有在某些情况下是可选的，但是，为了代码的一致性，建议都加上双引号。
- 避免为 0 值指定单位，例如，用 `margin: 0;` 代替 `margin: 0px;` 。

对于这里用到的术语有疑问吗？请参考 Wikipedia 上的 syntax section of the Cascading Style Sheets article。

```
/* Bad CSS */
.selector, .selector-secondary, .selector[type=text] {
  padding:15px;
  margin:0px 0px 15px;
  background-color:rgba(0, 0, 0, 0.5);
  box-shadow:0px 1px 2px #CCC,inset 0 1px 0 #FFFFFF
}

/* Good CSS */
.selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

声明顺序

相关的属性声明应当归为一组，并按照下面的顺序排列：

1. Positioning
2. Box model
3. Typographic
4. Visual

由于定位（positioning）可以从正常的文档流中移除元素，并且还能覆盖盒模型（box model）相关的样式，因此排在首位。盒模型排在第二位，因为它决定了组件的尺寸和位置。

其他属性只是影响组件的内部（*inside*）或者是不影响前两组属性，因此排在后面。

完整的属性列表及其排列顺序请参考 Recess。

```
.declaration-order {
  /* Positioning */
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 100;

  /* Box-model */
  display: block;
  float: right;
  width: 100px;
  height: 100px;

  /* Typography */
  font: normal 13px "Helvetica Neue", sans-serif;
  line-height: 1.5;
  color: #333;
  text-align: center;

  /* Visual */
  background-color: #f5f5f5;
  border: 1px solid #e5e5e5;
  border-radius: 3px;

  /* Misc */
  opacity: 1;
}
```

不要使用 `@import`

与 `<link>` 标签相比，`@import` 指令要慢很多，不光增加了额外的请求次数，还会导致不可预料的问题。替代办法有以下几种：

- 使用多个 `<link>` 元素
- 通过 Sass 或 Less 类似的 CSS 预处理器将多个 CSS 文件编译为一个文件
- 通过 Rails、Jekyll 或其他系统中提供过 CSS 文件合并功能

请参考 Steve Souders 的文章了解更多知识。

```
<!-- Use link elements -->
<link rel="stylesheet" href="core.css">

<!-- Avoid @imports -->
<style>
  @import url("more.css");
</style>
```

媒体查询（Media query）的位置

将媒体查询放在尽可能相关规则的附近。不要将他们打包放在一个单一样式文件中或者放在文档底部。如果你把他们分开了，将来只会被大家遗忘。下面给出一个典型的实例。

```
.element { ... }
.element-avatar { ... }
.element-selected { ... }

@media (min-width: 480px) {
  .element { ...}
  .element-avatar { ... }
  .element-selected { ... }
}
```

带前缀的属性

当使用特定厂商的带有前缀的属性时，通过缩进的方式，让每个属性的值在垂直方向对齐，这样便于多行编辑。

在 Textmate 中，使用 **Text → Edit Each Line in Selection** (??A)。在 Sublime Text 2 中，使用 **Selection → Add Previous Line** (??↑) 和 **Selection → Add Next Line** (??↓)。

```
/* Prefixed properties */
.selector {
  -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.15);
          box-shadow: 0 1px 2px rgba(0,0,0,.15);
}
```

单行规则声明

对于只包含一条声明的样式，为了易读性和便于快速编辑，建议将语句放在同一行。对于带有多条声明的样式，还是应当将声明分为多行。

这样做的关键因素是为了错误检测 -- 例如，CSS 校验器指出在 183 行有语法错误。如果是单行单条声明，你就不会忽略这个错误；如果是单行多条声明的话，你就要仔细分析避免漏掉错误了。

```
/* Single declarations on one line */
.span1 { width: 60px; }
.span2 { width: 140px; }
.span3 { width: 220px; }

/* Multiple declarations, one per line */
.sprite {
  display: inline-block;
  width: 16px;
  height: 15px;
  background-image: url(../img/sprite.png);
}
.icon           { background-position: 0 0; }
.icon-home      { background-position: 0 -20px; }
.icon-account   { background-position: 0 -40px; }
```

简写形式的属性声明

在需要显示地设置所有值的情况下，应当尽量限制使用简写形式的属性声明。常见的滥用简写属性声明的情况如下：

- `padding`
- `margin`
- `font`
- `background`
- `border`
- `border-radius`

大部分情况下，我们不需要为简写形式的属性声明指定所有值。例如，HTML 的 heading 元素只需要设置上、下边距（margin）的值，因此，在必要的时候，只需覆盖这两个值就可以。过度使用简写形式的属性声明会导致代码混乱，并且会对属性值带来不必要的覆盖从而引起意外的副作用。

MDN（Mozilla Developer Network）上一片非常好的关于shorthand properties 的文章，对于不太熟悉简写属性声明及其行为的用户很有用。

```
/* Bad example */
.element {
  margin: 0 0 10px;
  background: red;
  background: url("image.jpg");
  border-radius: 3px 3px 0 0;
}

/* Good example */
.element {
  margin-bottom: 10px;
  background-color: red;
  background-image: url("image.jpg");
  border-top-left-radius: 3px;
  border-top-right-radius: 3px;
}
```

Less 和 Sass 中的嵌套

避免非必要的嵌套。这是因为虽然你可以使用嵌套，但是并不意味着应该使用嵌套。只有在必须将样式限制在父元素内（也就是后代选择器），并且存在多个需要嵌套的元素时才使用嵌套。

```
// Without nesting
.table > thead > tr > th { … }
.table > thead > tr > td { … }

// With nesting
.table > thead > tr {
  > th { … }
  > td { … }
}
```

注释

代码是由人编写并维护的。请确保你的代码能够自描述、注释良好并且易于他人理解。好的代码注释能够传达上下文关系和代码目的。不要简单地重申组件或 class 名称。

对于较长的注释，务必书写完整的句子；对于一般性注解，可以书写简洁的短语。

```
/* Bad example */
/* Modal header */
.modal-header {
  ...
}

/* Good example */
/* Wrapping element for .modal-title and .modal-close */
.modal-header {
  ...
}
```

class 命名

- class 名称中只能出现小写字符和破折号（dashe）（不是下划线，也不是驼峰命名法）。破折号应当用于相关 class 的命名（类似于命名空间）（例如， `.btn` 和 `.btn-danger` ）。
- 避免过度任意的简写。 `.btn` 代表 *button*，但是 `.s` 不能表达任何意思。
- class 名称应当尽可能短，并且意义明确。
- 使用有意义的名称。使用有组织的或目的明确的名称，不要使用表现形式（presentational）的名称。
- 基于最近的父 class 或基本（base） class 作为新 class 的前缀。
- 使用 `.js-*` class 来标识行为（与样式相对），并且不要将这些 class 包含到 CSS 文件中。

在为 Sass 和 Less 变量命名是也可以参考上面列出的各项规范。

```
/* Bad example */
.t { ... }
.red { ... }
.header { ... }

/* Good example */
.tweet { ... }
.important { ... }
.tweet-header { ... }
```

选择器

- 对于通用元素使用 class ，这样利于渲染性能的优化。
- 对于经常出现的组件，避免使用属性选择器（例如， `[class^="..."]` ）。浏览器的性能会受到这些因素的影响。
- 选择器要尽可能短，并且尽量限制组成选择器的元素个数，建议不要超过 3 。
- 只有在必要的时候才将 class 限制在最近的父元素内（也就是后代选择器）（例如，不使用带前缀的 class 时 -- 前缀类似于命名空间）。

扩展阅读：

- Scope CSS classes with prefixes
- Stop the cascade

```
/* Bad example */
span { ... }
.page-container #stream .stream-item .tweet .tweet-header .username
.avatar { ... }

/* Good example */
.avatar { ... }
.tweet-header .username { ... }
.tweet .avatar { ... }
```

代码组织

- 以组件为单位组织代码段。
- 制定一致的注释规范。
- 使用一致的空白符将代码分隔成块，这样利于扫描较大的文档。
- 如果使用了多个 CSS 文件，将其按照组件而非页面的形式分拆，因为页面会被重组，而组件只会被移动。

```
/*
 * Component section heading
 */

.element { ... }

/*
 * Component section heading
 *
 * Sometimes you need to include optional context for the entire co
 */

.element { ... }

/* Contextual sub-component or modifer */
.element-heading { ... }
```

编辑器配置

将你的编辑器按照下面的配置进行设置，以避免常见的代码不一致和差异：

- 用两个空格代替制表符（soft-tab 即用空格代表 tab 符）。

- 保存文件时，删除尾部的空白符。
- 设置文件编码为 UTF-8。
- 在文件结尾添加一个空白行。

参照文档并将这些配置信息添加到项目的 `.editorconfig` 文件中。例如：
Bootstrap 中的 .editorconfig 实例。更多信息请参考 about EditorConfig。

W3School Foundation 教程

作者：W3School

来源：Foundation 教程

Foundation 入门

Foundation 5 简介



Foundation 用于开发响应式的 HTML, CSS and JavaScript 框架。

Foundation 是一个易用、强大而且灵活的框架,用于构建基于任何设备上的 Web 应用。

Foundation 是一个以移动优先的流行框架。

在线实例

菜鸟教程包含了上百个 Foundation 实例。

你可以直接使用我们的在线编辑器，并点击"提交运行"按钮查看结果：

Foundation 实例

```
<div class="row">
  <div class="medium-12 columns">
    <div class="panel">
      <h1>Foundation Page</h1>
      <p>Resize this responsive page to see the effect!</p>
      <button type="button" class="button small">I Like It!</button
    </div>
  </div>
</div>

<div class="row">
  <div class="medium-4 columns">
    <h3>Column 1</h3>
    <p>Lorem ipsum..</p>
  </div>
  <div class="medium-4 columns">
    <h3>Column 2</h3>
    <p>Lorem ipsum..</p>
  </div>
  <div class="medium-4 columns">
    <h3>Column 3</h3>
    <p>Lorem ipsum..</p>
  </div>
</div>
```

点击"尝试一下"按钮查看在线实例。

Foundation 特性

以下效果为在 iframe 标签中的演示，可以点击"尝试一下"查看在线实例：

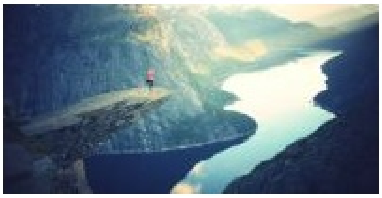
按钮

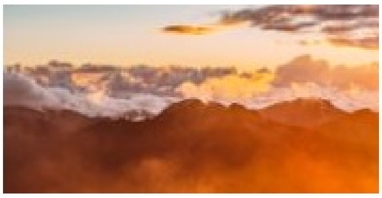


表格:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

图片弹窗:







警告:

绿色警告框 ×

红色警告框 ×

网格:

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

进度条:

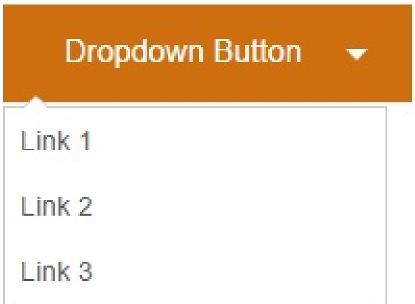
75%

50%

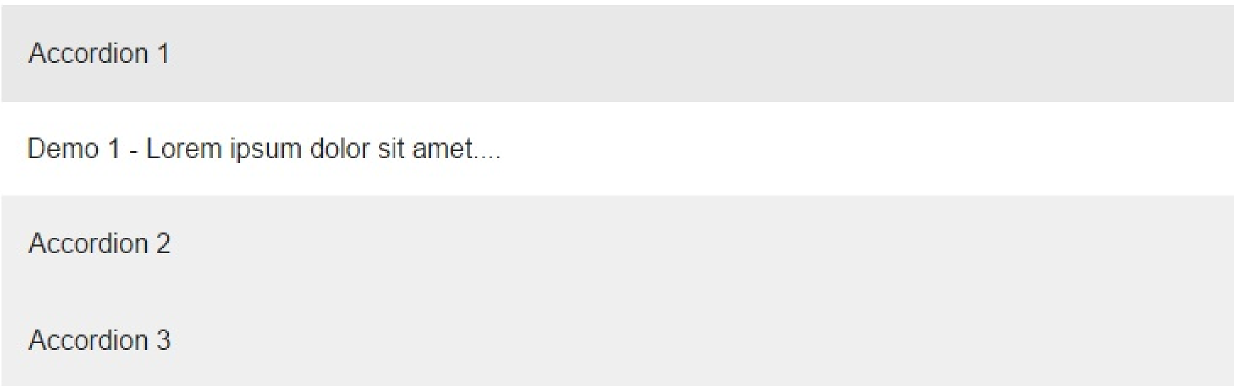
面板:



下拉菜单:



手风琴效果:

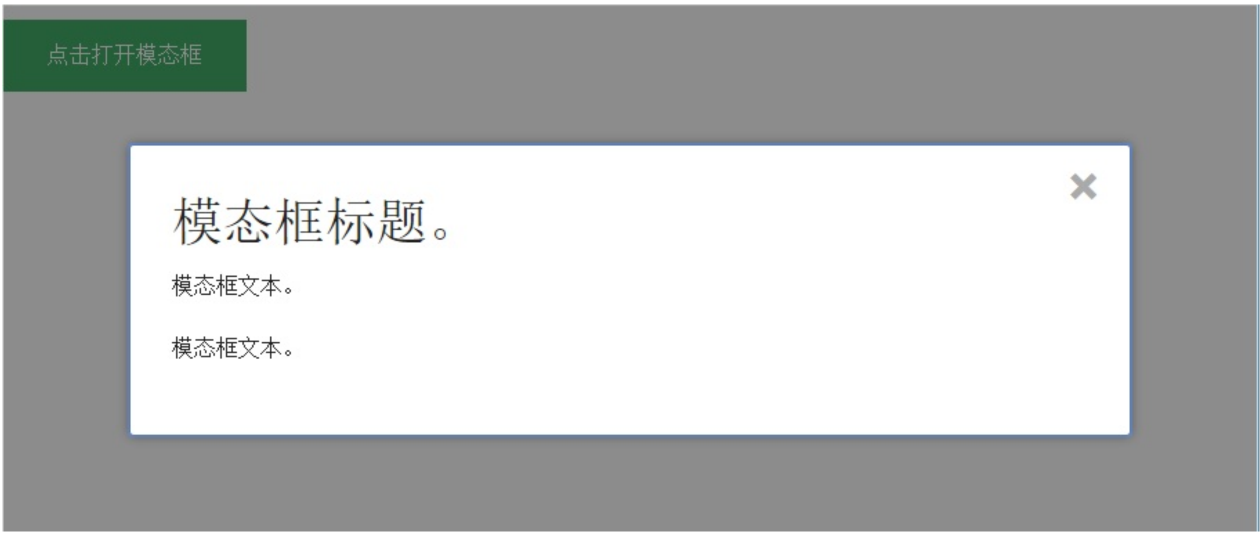


>

顶部导航:



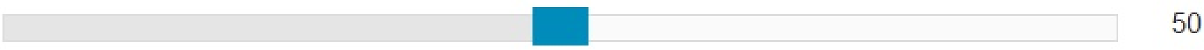
模态框:



开关:



滑块:



Foundation 起步

什么是 Foundation？

- Foundation 是一个免费的前端框架，用于快速开发。
- Foundation 包含了 HTML 和 CSS 的设计模板，提供多种 Web 上的 UI 组件，如表单、按钮、Tabs 等等。同时也提供了多种 JavaScript 插件。
- Foundation 移动优先，可创建响应式网页。
- Foundation 适用于初学者和专业人士。
- Foundation 已使用在 Facebook, eBay, Samsung, Amazon, Disney等。

> 什么是响应式网页设计？
>
> 响应式Web设计(Responsive Web design)的理念是：页面的设计与开发应当根据用户行为以及设备环境(系统平台、屏幕尺寸、屏幕定向等)进行相应的响应和调整。

从哪里下载 Foundation?

你可以通过以下两种方式来获取 Foundation：

1、从官网下载最新版本：http://foundation.zurb.com/。

2、使用菜鸟教程官网提供的CDN（推荐）：

```
<!-- css 文件 -->
<link rel="stylesheet" href="http://static.runoob.com/assets/founda

<!-- jQuery 库 -->
<script src="http://static.runoob.com/assets/jquery/2.0.3/jquery.mi

<!-- JavaScript 文件 -->
<script src="http://static.runoob.com/assets/foundation-5.5.3/js/fo

<!-- modernizr 文件 -->
<script src="http://static.runoob.com/assets/foundation-5.5.3/js/ve
```

本站静态 CDN 基于阿里云服务。

> **Foundation 使用 CDN 的优势:**
>
> Foundation 使用 CDN 提高了企业站点(尤其含有大量图片和静态页面站点)的访问速度，并大大提高以上性质站点的稳定性
>
> 为什么使用 **modernizr?**
>
> 一些 Foundation 的组件使用了比较前前沿的 HTML5 和 CSS3 特性，但不是所有浏览器都支持。 Modernizr 是一个用于检测用户浏览器HTML5和CSS3特性的JavaScript库 - 让组件能在所有浏览器上正常运行。

使用 Foundation 创建页面

1. 添加 HTML5 doctype

Foundation 使用 HTML 元素和 CSS 属性，所以需要添加 HTML5 doctype 文档类型声明。

同时我们可以设置文档的语言属性 lang 及字符编码：

```
<!DOCTYPE html>
<html lang="zh-cn">
  <head>
    <meta charset="utf-8">
  </head>
</html>
```

2. Foundation 5 移动优先

Foundation 5 为移动设备的响应式设计。框架的核心是移动优先。

为了确保页面可自由缩放可以在 <head> 元素中添加以下 <meta> 标签:

```
<meta name="viewport" content="width=device-width, initial-scale=1'
```

- width：控制 viewport 的大小，可以指定的一个值，如果 600，或者特殊的值，如 device-width 为设备的宽度（单位为缩放为 100% 时的 CSS 的像素）。
- initial-scale：初始缩放比例，也即是当页面第一次 load 的时候缩放比例。

3. 初始化组件

一些 Foundation 组件是基于 jQuery 开放的，如：模态框、下拉菜单等。你可以使用以下脚本来初始化组件：

```
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

基本 Foundation 页面

如何创建一个基本的 foundation 页面:

```
<!DOCTYPE html>
<html>
<head>
  <title>Foundation Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=
  <!-- css 文件 -->
  <link rel="stylesheet" href="http://static.runoob.com/assets/foun

  <!-- jQuery 库 -->
  <script src="http://static.runoob.com/assets/jquery/2.0.3/jquery.

  <!-- JavaScript 文件 -->
  <script src="http://static.runoob.com/assets/foundation-5.5.3/js/

  <!-- modernizr 文件 -->
  <script src="http://static.runoob.com/assets/foundation-5.5.3/js/
</head>
<body>

<div class="row">
  <div class="medium-12 columns">
    <div class="panel">
      <h1>Foundation 页面</h1>
      <p>重置窗口大小，查看效果！</p>
      <button type="button" class="button small">I Like It!</button
    </div>
  </div>
</div>

<div class="row">
  <div class="medium-4 columns">
    <h3>Column 1</h3>
    <p>Lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
  </div>
  <div class="medium-4 columns">
    <h3>Column 2</h3>
    <p>Lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
  </div>
  <div class="medium-4 columns">
    <h3>Column 3</h3>
    <p>Lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
  </div>
</div>

</body>
</html>
```

Foundation 文本

Foundation 默认设置

Foundation 使用浏览器默认字体大小 (`font-size:100%`)。对于大多数桌面浏览器来说，字体默认为 16px。对于大多数移动端浏览器来说，字体默认为 12px。默认的字体为 `"Helvetica Neue"` ， line-height 默认为 `1.5` 。

以上默认的设置均是针对 <body> 元素。

Foundation 文字排列设计

本章节我们将讨论 Foundation 的文字排列设计。

以下实例的真实样式请通过点击"尝试一下"按钮查看。

<h1> - <h6>

Foundation 渲染的 HTML 标题 (<h1> 到 <h6>) 如下所示:

实例

```
<h1>h1 标题</h1>
<h2>h2 标题</h2>
<h3>h3 标题</h3>
<h4>h4 标题</h4>
<h5>h5 标题</h5>
<h6>h6 标题</h6>
```

提示： 如果需要创建一个浅色的标题，可以在元素上添加 `.subheader` 类:

实例

```
<h1 class="subheader">h1.subheader</h1>
<h2 class="subheader">h2.subheader</h2>
<h3 class="subheader">h3.subheader</h3>
<h4 class="subheader">h4.subheader</h4>
<h5 class="subheader">h5.subheader</h5>
<h6 class="subheader">h6.subheader</h6>
```

<small>

在 Foundation 中， HTML `<small>` 元素用于创建一个浅色的副标题：

实例

```
<h1>h1 heading <small>secondary text</small></h1>
<h2>h2 heading <small>secondary text</small></h2>
<h3>h3 heading <small>secondary text</small></h3>
<h4>h4 heading <small>secondary text</small></h4>
<h5>h5 heading <small>secondary text</small></h5>
<h6>h6 heading <small>secondary text</small></h6>
```

<a>

Foundation `<a>` 元素的样式如下所示：

实例

```
<p>这是一个 <a class="a" href="#">链接</a>。</p>
```

<abbr>

Foundation `<abbr>` 元素的样式如下所示：

实例

```
<p>The <abbr title="World Health Organization">WHO</abbr> was found
```

<blockquote>

Foundation `<blockquote>` 元素的样式如下所示：

实例

```
<blockquote>
  <p>学的不仅是技术，更是梦想！</p>
  <cite>菜鸟教程</cite>
</blockquote>
```

<dl>

Foundation `<dl>` 元素的样式如下所示：

实例

```
<dl>
        <dt>Coffee</dt>
        <dd>- black hot drink</dd>
        <dt>Milk</dt>
        <dd>- white cold drink</dd>
      </dl>
```

<code>

Foundation `<code>` 元素的样式如下所示：

实例

```
<p>以下 HTML 元素: <code>span</code>, <code>section</code>, 和 <code
```

<kbd>

Foundation `<kbd>` 元素的样式如下所示：

实例

```
<p>按下 <kbd>ctrl + p</kbd> 键打开打印窗口。</p>
```

<hr>

Foundation `<hr>` 元素的样式如下所示：

实例

```
<hr>
```

Foundation 表格

Foundation 的 `<table>` 元素样式为灰色斑马条纹且包含四个边框：

Firstname	Lastname	Email	City	Age	Country
John	Doe	john@example.com	New York	35	USA
Mary	Moe	mary@example.com	Chicago	51	USA
July	Dooley	july@example.com	San Francisco	38	USA

实例

```
<table>
  <thead>
    <tr>
      <th>Firstname</th>
      <th>Lastname</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>Doe</td>
      <td>john@example.com</td>
    </tr>
    <tr>
      <td>Mary</td>
      <td>Moe</td>
      <td>mary@example.com</td>
    </tr>
    <tr>
      <td>July</td>
      <td>Dooley</td>
      <td>july@example.com</td>
    </tr>
  </tbody>
</table>
```

响应式表格

使用 CSS 让表格支持响应式设计：在表格外添加 `<div>` 元素，样式为 `overflow-x:hidden`：

实例

```
<div style="overflow-x:hidden">
  <table>
    ...
  </table>
</div>
```

Foundation 按钮

按钮样式

Foundation 提供了 6 种按钮样式。 `.button` 类创建了一个蓝色的按钮并附有内边距。不同颜色按钮类为： `.secondary` , `.success` , `.info` , `.warning` 或 `.alert` :

实例

```
<button type="button" class="button">Default</button>
<button type="button" class="button secondary">Secondary</button>
<button type="button" class="button success">Success</button>
<button type="button" class="button info">Info</button>
<button type="button" class="button warning">Warning</button>
<button type="button" class="button alert">Alert</button>
```

按钮类可以使用在 `<a>` , `<button>` , 或 `<input>` 元素:

实例

```
<a href="#" class="button info" role="button">Link Button</a>
<button type="button" class="button info">Button</button>
<input type="button" class="button info" value="Input Button">
<input type="submit" class="button info" value="Submit Button">
```

> **为什么将 a 标签的 href 设置为 # ?**
>
> 当我们不希望链接点击跳转并得到 "404" 页面时，我们可以将 a 标签的 href 设置为 #。

按钮大小

我们可以使用 `.large` , `.small` 或 `.tiny` 类来设置按钮大小：

实例

```
<button type="button" class="button large">Large</button>
<button type="button" class="button">Default</button>
<button type="button" class="button small">Small</button>
<button type="button" class="button tiny">Tiny</button>
```

圆角按钮

可以使用 `.radius` 和 `.round` 类来设置圆角按钮：

实例

```
<button type="button" class="button">Default Button</button>
<button type="button" class="button radius">Radius Button</button>
<button type="button" class="button round">Round Button</button>
```

延展按钮

可以使用 `.expand` 类来设置按钮的宽带为 100%:

实例

```
<button type="button" class="button">Default Button</button>
<button type="button" class="button expand">Expanded Button</button
```

禁用按钮

可以使用 `.disabled` 类来设置按钮不可点击:

实例

```
<button type="button" class="button">Default Button</button>
<button type="button" class="button disabled">Disabled Button</butt
```

Foundation 按钮组

按钮组

Foundation 可以在同一行内创建一系列的按钮。

可以使用 `<ul>` 元素并添加 `.button-group` 类来创建按钮组：

实例

```
<ul class="button-group">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
</ul>
```

垂直按钮组

垂直按钮组使用 `.stack` 类来创建。注意，按钮会跨越父元素的整个宽度:

实例

```
<ul class="button-group stack">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
</ul>
```

`.stack-for-small` 类用于小尺寸的屏幕，按钮有水平排列变为垂直排列：

实例

```
<ul class="button-group stack-for-small">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
</ul>
```

圆角按钮组

按钮组中使用 `.radius` 和 `.round` 类为按钮添加圆角效果：

实例

```
<ul class="button-group radius">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
</ul>

<ul class="button-group round">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
</ul>
```

均匀延展按钮组

`.even-num` 类用于在按钮组中均匀的分配按钮的宽度并跨越父元素 100% 宽度。*num* 为按钮组中按钮的数量，从 1 到 8：

实例

```
<ul class="button-group even-3">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
</ul>

<ul class="button-group even-5">
  <li><button type="button" class="button">Apple</button></li>
  <li><button type="button" class="button">Samsung</button></li>
  <li><button type="button" class="button">Sony</button></li>
  <li><button type="button" class="button">HTC</button></li>
  <li><button type="button" class="button">Huawei</button></li>
</ul>

<ul class="button-group even-8">
  <li><button type="button" class="button">A</button></li>
  <li><button type="button" class="button">B</button></li>
  <li><button type="button" class="button">C</button></li>
  <li><button type="button" class="button">D</button></li>
  <li><button type="button" class="button">E</button></li>
  <li><button type="button" class="button">F</button></li>
  <li><button type="button" class="button">G</button></li>
  <li><button type="button" class="button">H</button></li>
</ul>
```

下拉菜单按钮

下拉菜单按钮可以让用户选取设定好的值：

实例

```
<!-- Trigger the dropdown -->
<a href="#" data-dropdown="id01" class="button dropdown">Dropdown E

<!-- The actual dropdown -->
<ul id="id01" data-dropdown-content class="f-dropdown">
  <li><a href="#">Link 1</a></li>
  <li><a href="#">Link 2</a></li>
  <li><a href="#">Link 3</a></li>
</ul>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

实例解析

`.dropdown` 类创建一个下拉菜单按钮。

使用带有 `data-dropdown="_id_"` 属性的按钮或链接打开下拉菜单。

id 值需要与下拉菜单的内容 (id01) 匹配。

在 `<ul>` 中添加 `.f-dropdown` 类和 `data-dropdown-content` 属性来创建下拉菜单的内容。

最后初始化 Foundation JS。

分割按钮

我们也可以创建一个分割按钮的下拉菜单。只需要在按钮中添加 `.split` 类并使用 span 元素生成一个方向箭的按钮：

实例

```
<button class="button split">Split Button
  <span data-dropdown="id01"></span>
</button>

<ul id="id01" data-dropdown-content class="f-dropdown">
  <li><a href="#">Link 1</a></li>
  <li><a href="#">Link 2</a></li>
  <li><a href="#">Link 3</a></li>
</ul>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

> 提示:后面的教程中我们将学到更多关于下拉菜单是知识。

Foundation 图标

Foundation 提供了 283 个图标，你可以使用css来定义它的样式尺寸。

图标可用于文本，按钮，工具条，导航栏，表单等。

以下是 Foundation 图标的实例：

刷新按钮:

检测图标:

主页图标:

图标语法

创建图标语法格式如下:

```
<i class="fi-name"></i>
```

name 部分替换为图标的名字。

要使用图标我们需要在 <head> 部分添加图标的样式文件：

```
<link rel="stylesheet" href="http://static.runoob.com/assets/founda
```

Icon 实例

以下演示了使用图标的实例:

实例

```
<p>Cloud icon: <i class="fi-cloud"></i></p>
<p>Cloud icon as a link:
  <a href="#"><i class="fi-cloud"></i></a>
</p>
<p>Styled Cloud icon: <i class="fi-cloud" style="font-size:35px;col
<p>Home icon: <i class="fi-home"></i></p>
<p>Home icon on a button:
  <button type="button" class="button">
    <i class="fi-home"></i> Home
  </button>
</p>
<p>Home icon on a green button:
  <button type="button" class="button success">
    <i class="fi-home"></i> Home
  </button>
</p>
<p>Home icon on a large, light blue link button:
  <a href="#" class="button info large">
    <i class="fi-home"></i> Home
  </a>
</p>
```

工具条图标

我们可以使用 `.icon-bar` 类来创建一个指定数量的工具栏图标:

实例

```
<div class="icon-bar five-up">
  <a href="#" class="item">
    <i class="fi-home"></i>
  </a>
  <a href="#" class="item">
    <i class="fi-bookmark"></i>
  </a>
  <a href="#" class="item">
    <i class="fi-info"></i>
  </a>
  <a href="#" class="item">
    <i class="fi-mail"></i>
  </a>
  <a href="#" class="item">
    <i class="fi-like"></i>
  </a>
</div>
```

图标描述使用 `<label>` 元素:

实例

```
<div class="icon-bar five-up">
  <a href="#" class="item">
    <i class="fi-home"></i>
    <label>Home</label>
  </a>
  <a href="#" class="item">
    <i class="fi-share"></i>
    <label>Share</label>
  </a>
  <a href="#" class="item">
    <i class="fi-info"></i>
    <label>Info</label>
  </a>
  <a href="#" class="item">
    <i class="fi-mail"></i>
    <label>Mail</label>
  </a>
  <a href="#" class="item">
    <i class="fi-magnifying-glass"></i>
    <label>Search</label>
  </a>
</div>
```

`.disabled` 类可以让图标变成不可点击状态：

实例

```
<a href="#" class="item disabled">
  <i class="fi-share"></i>
</a>

<a href="#" class="item disabled">
    <i class="fi-mail"></i>
</a>
```

`.vertical` 类用于创建一个垂直的工具栏：

实例

```
<div class="icon-bar vertical five-up">
  ....
</div>
```

Foundation 图标参考手册

更多关于图标的内容，可以参考我们的 Foundation 图标手册。

Foundation 标签

`.label` 类用于提供一些附加信息:

实例

```
<h2>Example <span class="label">New</span></h2>
<h3>Example <span class="label">New</span></h3>
<h4>Example <span class="label">New</span></h4>
```

以下类可以设置标签的颜色: `.secondary` , `.success` , `.info` , `.warning` 或 `.alert` :

实例

```
<span class="label">Default Label</span>
<span class="label secondary">Secondary Label</span>
<span class="label success">Success Label</span>
<span class="label info">Info Label</span>
<span class="label warning">Success Label</span>
<span class="label alert">Alert Label</span>
```

圆角标签

`.radius` 与 `.round` 类可以为标签添加圆角：

实例

```
<span class="label">Default Label</span>
<span class="label radius">Radius Label</span>
<span class="label round">Round Label</span>
<span class="label success round">5</span>
```

标签大小

可以使用 CSS 来修改标签的大小:

实例

```
<h1>Example <span class="label" style="font-size:36px;">New</span><
<h2>Example <span class="label" style="font-size:30px;">New</span><
<h3>Example <span class="label" style="font-size:20px;">New</span><
<h4>Example <span class="label">New</span></h4>
```

Foundation 提醒框

Foundation 可以很简单的创建一个提醒框：



提醒框可以使用 `.alert-box` 类创建, 可以添加可选的类： `.secondary` , `.success` , `.info` , `.warning` 或 `.alert` :

实例

```
<div data-alert class="alert-box">
  This is a default alert box.
</div>

<div data-alert class="alert-box secondary">
  This is a secondary alert box.
</div>

<div data-alert class="alert-box success">
  <strong>Success!</strong> This alert box indicates a successful c
</div>

<div data-alert class="alert-box info">
  <strong>Info!</strong> This alert box indicates a neutral informa
</div>

<div data-alert class="alert-box warning">
  <strong>Warning!</strong> This alert box indicates a warning that
</div>

<div data-alert class="alert-box alert">
  <strong>Alert!</strong> This alert box indicates a dangerous or p
</div>
```

> 提醒框的宽度为容器的 100%。

圆角提醒框

`.radius` 和 `.round` 类用于为提醒框添加圆角：

实例

```
<div data-alert class="alert-box success radius">
  <strong>Success!</strong> Alert box with a radius.
</div>

<div data-alert class="alert-box info round">
  <strong>Info!</strong> Alert box that is rounded.
</div>
```

关闭提醒框

要关闭提醒框，可以在连接或按钮元素上添加 `class="close"` 类，并初始化 Foundation JS:

实例

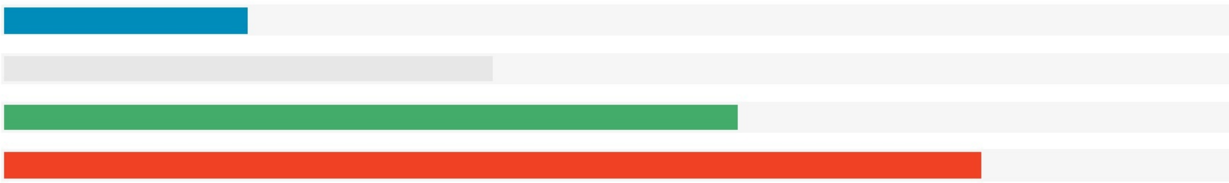
```
<div data-alert class="alert-box">
  This is a default alert box with closing functionality.
  <a href="#" class="close">&times;</a>
</div>

<script>
// Initialize Foundation JS For Functionality
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

> × (×) 是一个 HTML 字符实体表示一个关闭按钮的图标，而不是字母 "x"。

Foundation 进度条

Foundation 进度条可以作为程序处理的程度来显示：



我们可以在 `<div>` 元素中使用 `.progress` 类来创建进度条，`.meter` 类用于子元素(<span>)。我们可以在 <span> 元素中设置进度百分比，如下所示：

实例

```
<div class="progress">
  <span class="meter" style="width:70%"></span>
</div>
```

进度条颜色

默认情况下，进度条颜色为蓝色。不同颜色的类为：`.secondary`，`.success`，或 `.alert`：

实例

```
<div class="progress">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress secondary">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress success">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress alert">
  <span class="meter" style="width:50%"></span>
</div>
```

圆角进度条

`.radius` 和 `.round` 类用于为进度条添加圆角效果：

实例

```
<div class="progress">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress radius">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress rounded">
  <span class="meter" style="width:50%"></span>
</div>
```

进度条尺寸

可以使用 `.small-_num_` 或 `.large-_num_` 来修改进度条的宽度， *num* 是一个数字在 1(8.33%) 和 12(default (100%)) 之间:

实例

```
<div class="progress large-1">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress large-6">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress large-9">
  <span class="meter" style="width:50%"></span>
</div>

<div class="progress large-11">
  <span class="meter" style="width:50%"></span>
</div>

<!-- Default width -->
<div class="progress large-12">
  <span class="meter" style="width:50%"></span>
</div>
```

进度条标签

可以使用 CSS 为进度条添加标签。以下实例中我们添加了<span> 元素来显示百分比：

实例

```
 <style>
.percentage {
  position: absolute;
  top: 50%;
  left: 50%;
  color: white;
  transform: translate(-50%, -50%);
  font-size: 12px;
}
</style>

<div class="progress">
  <span class="meter" style="position:relative;width:75%">
    <span class="percentage">75%</span>
  </span>
</div>

<div class="progress success">
  <span class="meter" style="position:relative;width:50%">
    <span class="percentage">50%</span>
  </span>
</div>

<div class="progress alert">
  <span class="meter" style="position:relative;width:25%">
    <span class="percentage">25%</span>
  </span>
</div>
```

Foundation 面板

Foundation 面板是一个灰色边框，内容含有内边距的模块。可以使用 `.panel` 类来创建:

实例

```
<div class="panel">
  <h3>标题</h3>
  <p>文本内容..</p>
</div>
```

面板颜色

使用 `.callout` 类将面板颜色修改为浅蓝：

实例

```
<div class="panel callout">
  <h3>标题</h3>
  <p>文本内容..</p>
</div>
```

圆角面板

使用 `.radius` 类将面板设置为圆角：

实例

```
<div class="panel radius">
  <h3>标题</h3>
  <p>文本内容..</p>
</div>
```

自定义面板

可以使用 CSS 来自定义面板，以下实例中我们将面板作为一个卡片：

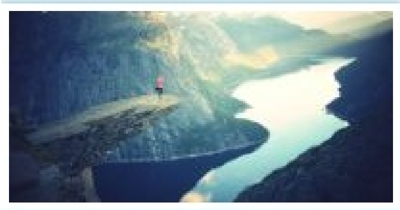
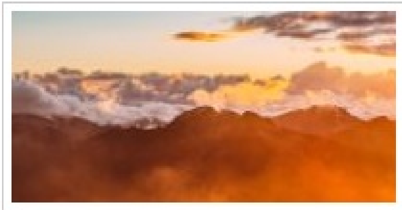
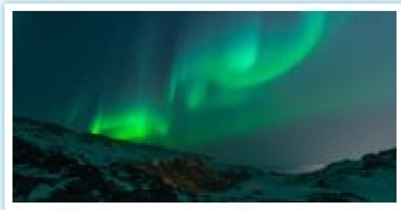
实例

```
<style>
.panel {
    padding: 0;
    border: none;
    width: 50%;
}
div.container {
    text-align: center;
    padding: 15px;
    margin-top: 20px;
}
img {
    width: 100%;
}
</style>

<div class="panel">
  <img src="img/20121204024112919.jpg" alt="Cinque Terre">
  <div class="container">
    <h4>长城</h4>
    <p>不到长城非好汉！！！</p>
  </div>
</div>
```

Foundation 图片

Foundation 提供了响应式的图片，可以创建缩略图喝图片弹窗：

缩略图

在 `<img>` 元素外添加 `<a>` 元素将图片作为一个锚链接。

在 `<a>` 标签中添加 `.th` 类将图片设置为缩略图。 鼠标移动到上面会显示一个浅蓝色外框:

实例

```
<a href="paris.jpg" class="th">
  <img src="paris.jpg" alt="Paris">
</a>
```

> 响应式图片
>
> Foundation 中图片默认是响应式的。我们可以在实例页面重置浏览器大小来查看图片缩放效果。

圆角图片

我们可以在 `.th` 类添加 `.radius` 类来设置圆角缩略图：

实例

```
<a href="paris.jpg" class="th radius">
  <img src="paris.jpg" alt="Paris">
</a>
```

简洁的弹窗

Foundation 可以很容易实现图片弹窗。

要创建一个弹窗可以在 `<ul>` 元素上添加 `.clearing-thumbs` 类及 `data-clearing` 属性。在 `<ul>` 内添加图片列表。

注意: 图片弹窗需要 JavaScript。所以使用它前需要初始化 Foundation JS。

实例

```
<ul class="clearing-thumbs" data-clearing>
  <li><a href="rock600x400.jpg" class="th"><img src="rock200x100.jp
  <li><a href="skies600x400.jpg" class="th"><img src="skies200x100
  <li><a href="lights600x400.jpg" class="th"><img src="lights200x10
</ul>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

图片文本描述

可以添加 `data-caption` 属性到每个图片来设置图片的描述:

实例

```
<ul class="clearing-thumbs" data-clearing>
  <li><a href="rock600x400.jpg" class="th"><img data-caption="The F
  <li><a href="skies600x400.jpg" class="th"><img data-caption="Sunn
  <li><a href="lights600x400.jpg" class="th"><img data-caption="Nor
</ul>
```

> 提示： 你可以在 data-caption 属性中添加 HTML 元素，如 data-caption="<h2>Pulpit Rock</h2><p>Located in Norway</p>"

只显示一张缩略图

当你需要实现只显示一张缩略图时你可以在 `<ul>` 中使用 `.clearing-feature` 类并在 `<li>` 中使用 `.clearing-featured-img` 类。

实例

```
<ul class="clearing-thumbs clearing-feature" data-clearing>
  <li><a href="rock600x400.jpg" class="th"><img data-caption="The R
  <li><a href="skies600x400.jpg" class="th"><img data-caption="Sunn
  <li class="clearing-featured-img"><a href="lights600x400.jpg" cla
</ul>
```

Foundation 下拉菜单

Foundation 下拉菜单允许用户从预定义的下拉列表中选取一个值：

实例

```
<!-- Trigger the Dropdown -->
<a href="#" data-dropdown="id01" class="button dropdown">Dropdown E

<!-- Dropdown content -->
<ul id="id01" data-dropdown-content class="f-dropdown">
  <li><a href="#">Link 1</a></li>
  <li><a href="#">Link 2</a></li>
  <li><a href="#">Link 3</a></li>
</ul>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

实例解析

`.dropdown` 类为按钮添加一个向下的箭头符号"图标。

使用按钮或链接的 `data-dropdown="_id_"` 属性来打开下拉菜单。

id 值需要与下拉菜单的内容 (id01) 匹配。

在 <div>, <nav>, `<ul>` 中添加 `.f-dropdown` 类和 `data-dropdown-content` 属性来创建下拉菜单的内容。

最后初始化 Foundation JS。

注意: 在小屏幕上，所有的下拉菜单的宽度是100%。

下拉菜单尺寸

使用 `.tiny` , `.small` , `.medium` , `.large` 或 `.mega` 来修改下拉菜单的宽度。

注意: 在小屏幕上，所有的下拉菜单的宽度是100%。

实例

```
<!-- Tiny Dropdown: max-width is 200px -->
<ul id="id01" data-dropdown-content class="f-dropdown tiny">..

<!-- Small Dropdown: max-width is 300px -->
<ul id="id02" data-dropdown-content class="f-dropdown small">..

<!-- Medium Dropdown: max-width is 500px -->
<ul id="id03" data-dropdown-content class="f-dropdown medium">

<!-- Large Dropdown: max-width is 800px -->
<ul id="id04" data-dropdown-content class="f-dropdown large">..

<!-- Mega Dropdown: 100% width -->
<ul id="id04" data-dropdown-content class="f-dropdown mega">..
```

下拉菜单边距

可以使用 `.content` 类为下拉菜单添加内边距：

实例

```
<!-- Default Dropdown -->
<ul id="id01" data-dropdown-content class="f-dropdown">..

<!-- Dropdown with padding -->
<ul id="id02" data-dropdown-content class="f-dropdown content">..
```

其他实例

<div> 下拉菜单中添加多媒体元素：

实例

```
<a href="#" data-dropdown="id01" class="button dropdown">Dropdown E
<div id="id01" data-dropdown-content class="f-dropdown medium conte
  <h4>Paris Title</h4>
  <p>Some text.. some text..</p>
  <img src="paris.jpg" alt="Paris" width="400" height="300">
  <p>Paris, je t'aime.</p>
</div>
```

下拉菜单方向

默认情况下下拉菜单在底部，可以通过添加 `data-options="align:left|right|top"` 来修改其方向：

实例

```
<a href="#" data-dropdown="id01" data-options="align:right" class='
<a href="#" data-dropdown="id02" data-options="align:top" class="bu
<a href="#" data-dropdown="id03" data-options="align:bottom" class=
<a href="#" data-dropdown="id04" data-options="align:left" class="b
```

下拉菜单触发条件

默认情况下，下拉菜单在点击按钮后显示。如果你需要在鼠标移动上去后显示，可以在按钮上使用 `data-options="is_hover:true"` 属性:

实例

```
<a href="#" data-dropdown="id01" data-options="is_hover:true" class
<ul id="id01" data-dropdown-content class="f-dropdown">
  <li><a href="#">Link 1</a></li>
  <li><a href="#">Link 2</a></li>
  <li><a href="#">Link 3</a></li>
</ul>
```

分割按钮

我们可以在按钮上添加 `.split` 类来设置一个分割效果的按钮，分割后会在 <span> 元素上生成一个方向向下的图标按钮:

实例

```
<button class="button split">Split Button
  <span data-dropdown="id01"></span>
</button>

<ul id="id01" data-dropdown-content class="f-dropdown">
  <li><a href="#">Link 1</a></li>
  <li><a href="#">Link 2</a></li>
  <li><a href="#">Link 3</a></li>
</ul>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

Foundation 折叠列表

在你想隐藏部分内容的显示时，可以使用折叠列表。

实例

```
<ul class="accordion" data-accordion>
  <li class="accordion-navigation">
    <a href="#demo">Simple Collapsible</a>
    <div id="demo" class="content">
      菜鸟教程 -- 学的不仅是技术，更是梦想！！！
    </div>
  </li>
</ul>

<!-- 初始化 Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

实例解析

`.accordion` 类和 `data-accordion` 属性用于创建一个可折叠的元素。 `.accordion-navigation` 类用于渲染可折叠元素。实际的内容在 `.content` 类 (<div class="content">) 中，点击按钮既可以显示。

我们在列表项中添加 `<a>` 元素来控制（显示/隐藏）可折叠的内容。然后锚链接使用与可折叠内容内容相同的id (<a href=#demo" 与 <div id="demo"> 相关联)。

注意： 可折叠的效果需要初始化 Foundation JS。

默认情况下，可折叠内容是隐藏的。我们可以通过在 `<div>` 上添加 `.active` 类让其默认是显示的：

实例

```
<div id="demo" class="content active">
```

手风琴效果

手风琴效果用于延展与设置可折叠内容。

手风琴效果通过使用多个不同的锚链接与id来创建：

实例

```
<ul class="accordion" data-accordion>
  <li class="accordion-navigation">
    <a href="#demo">手风琴实例 1</a>
    <div id="demo" class="content active">
      Demo 1 - 菜鸟教程 -- 学的不仅是技术，更是梦想！！！
    </div>
  </li>
  <li class="accordion-navigation">
    <a href="#demo2">手风琴实例 2</a>
    <div id="demo2" class="content">
      Demo 2 - Lorem ipsum dolor sit amet....
    </div>
  </li>
  <li class="accordion-navigation">
    <a href="#demo3">手风琴实例 3</a>
    <div id="demo3" class="content">
      Demo 3 - 菜鸟教程 -- 学的不仅是技术，更是梦想！！！
    </div>
  </li>
</ul>
```

默认情况下，手风琴列表项有一个是打开的。如果你想关闭所有可以使用 `data-options="multi_expand:true;"` 属性:

实例

```
<ul class="accordion" data-accordion data-options="multi_expand:tru
  ..
</ul>
```

Foundation 列表

在 HTML 中，无序列表 (`<ul>`) 实例如下:

```
<ul>
  <li>List item</li>
  <li>List item</li>
  <li>List item</li>
  <li>List item</li>
</ul>
```

结果:

- List item
- List item
- List item
- List item

圆圈标识符

在 Foundation 中，无序列表 (`<ul>`) 的前缀符号为圆圈，使用 .circle 类，实例如下:

```
<ul class="circle">
  <li>List item</li>
  ...
</ul>
```

方块标识符

方块标识符前缀使用 .square 类:

```
<ul class="square">
  <li>List item</li>
  ...
</ul>
```

无标识符

如果你不需要标识符，你可以使用 .no-bullet 类 :

```
<ul class="no-bullet">
  <li>List item</li>
  ...
</ul>
```

如果你需要添加一个水平的列表，可以在 `<ul>` 上添加 `.list-inline` 类：

```
<ul class="inline-list">
```

列表菜单

一些 Web 页面可能需要列表菜单。

在 HTML 中，列表菜单同无序列表 `<ul>` 来定义，例如:

```
<ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">Menu 1</a></li>
  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>
</ul>
```

结果:

- Home
- Menu 1
- Menu 2
- Menu 3

如果你需要添加一个水平的菜单，可以在 `<ul>` 上添加 `.list-inline` 类：

```
<ul class="inline-list">
```

Foundation 选项卡

选项卡导航可以很好的展示不同的内容，并可以对内容进行切换。



选项卡使用 `<ul class="tabs" data-tab>` 来创建, 各个选项使用 <li> 元素并加上 `.tab-title` 类来创建。

提示: 当前选中项可以使用 `.active` 类。

实例

```
<ul class="tabs" data-tab>
  <li class="tab-title active"><a href="#">Home</a></li>
  <li class="tab-title"><a href="#">Menu 1</a></li>
  <li class="tab-title"><a href="#">Menu 2</a></li>
  <li class="tab-title"><a href="#">Menu 3</a></li>
</ul>
```

垂直的选项卡

垂直选项卡可以使用 `.vertical` 类:

实例

```
<ul class="tabs vertical" data-tab>
```

切换选项卡

如果要设置切换标签，可以使用 <div> 元素加上 `.content` 类。每个选项卡上加上唯一的 ID, 并连接到列表项 (<a href="#menu1" 将打开 id="menu1" 的选项内容)。最后将所有的选项内容放在 `<div>` 元素上，该 `<div>` 元素需要添加 `.tabs-content` 类，并初始化 Foundation JS。

注意 `.active` 类会自动添加到当前选中的选项卡上：

实例

```
<ul class="tabs" data-tab>
  <li class="tab-title active"><a href="#home">Home</a></li>
  <li class="tab-title"><a href="#menu1">Menu 1</a></li>
  <li class="tab-title"><a href="#menu2">Menu 2</a></li>
  <li class="tab-title"><a href="#menu3">Menu 3</a></li>
</ul>
<div class="tabs-content">
  <div class="content active" id="home">
    <h3>HOME</h3>
    <p>Home is where the heart is..</p>
  </div>
  <div class="content" id="menu1">
    <h3>Menu 1</h3>
    <p>Some text, blabla</p>
  </div>
  <div class="content" id="menu2">
    <h3>Menu 2</h3>
    <p>Some other text.</p>
  </div>
  <div class="content" id="menu3">
    <h3>Menu 3</h3>
    <p>Last tab.</p>
  </div>
</div>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

选项卡淡入

使用 CSS 来自定义选项卡淡入的效果：

实例

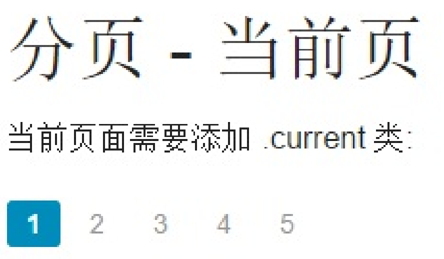
```
.tabs-content > .content.active {
    -webkit-animation: fadeEffect 1s;
    animation: fadeEffect 1s;
}

@-webkit-keyframes fadeEffect {
    from {opacity: 0;}
    to {opacity: 1;}
}

@keyframes fadeEffect {
    from {opacity: 0;}
    to {opacity: 1;}
}
```

Foundation 分页

如果你的网页有很多内容，就需要使用分页功能。



要创建一个基础的分页功能需要在 `<ul>` 元素上加上 `.pagination` 类:

实例

```
<ul class="pagination">
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
</ul>
```

当前页面

可以在 `<li>` 加上 `.current` 类来标注当前页面：

实例

```
<ul class="pagination">
  <li class="current"><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
</ul>
```

禁用分页

如果需要设置某个分页不可点击需要使用 `.unavailable` 类：

实例

```
<ul class="pagination">
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li class="unavailable"><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
</ul>
```

分页方向

在第一个和最后一个 code> 元素上添加 `.arrow` 类插入 HTML 实体符号 `«` 和 `»` 来创建分页方向符号：

实例

```
<ul class="pagination">
  <li class="arrow"><a href="#">&laquo;</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li class="arrow"><a href="#">&raquo;</a></li>
</ul>
```

分页居中显示

我们可以在 <ul> 外层添加 `<div>` 元素，并在 `<div>` 上添加 `.pagination-centered` 类来实现分页居中显示：

实例

```
<div class="pagination-centered">
  <ul class="pagination">
    <li class="arrow"><a href="#">&laquo;</a></li>
    <li class="current"><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
    <li class="arrow"><a href="#">&raquo;</a></li>
  </ul>
</div>
```

面包屑导航

面包屑导航用于展示当前页面的导航结构。

在 <ul> 元素上添加 `.breadcrumbs` 类来实现面包屑导航。你可以在 <li> 上添加 `.current` 或 `.unavailable` 类设置当前页与不可点击效果:

实例

```
<ul class="breadcrumbs">
  <li><a href="#">Home</a></li>
  <li><a href="#">Private</a></li>
  <li class="unavailable"><a href="#">Pictures</a></li>
  <li class="current">Vacation</li>
</ul>
```

子导航

在页面切换上，子导航是非常有用的。

在 <dl> 元素上添加 `.sub-nav` 类来创建子导航。在 <dt> 元素上添加标题，为选中的选项 <dd> 添加 `.active` 类:

实例

```
<ul class="sub-nav">
  <dt>Filter:</dt>
  <dd class="active"><a href="#">All</a></dd>
  <dd><a href="#">Active</a></dd>
  <dd><a href="#">Pending</a></dd>
  <dd><a href="#">Suspended</a></dd>
</ul>
```

Foundation 价格表

价格表可用于展示企业产品:

实例

```
<ul class="pricing-table">
  <li class="title">Basic</li>
  <li class="price">$19.99</li>
  <li class="description">Great for small business</li>
  <li class="bullet-item">24/7 Support</li>
  <li class="bullet-item">15GB Storage</li>
  <li class="bullet-item">1GB Bandwidth</li>
  <li class="cta-button"><a class="button" href="#">Buy</a></li>
</ul>
```

实例解析

- `ul.pricing-table` - 定义价格表
- `li.title` - 定义产品标题 (黑色背景)
- `li.price` - 定义价格 (灰色背景字体大个项)
- `li.description` - 定义产品描述 (如果需要)
- `li.bullet-item` - 定义产品特点
- `li.ca-button` - 按钮文本 (如 "Buy", "Join", "Sign Up", 等)

注意: 表格会 100% 填充容器的宽度，所有的项都有边框且是居中的。

价格表网格

以下实例显示了三个企业产品的价格（三项是均等划分宽度的）：

实例

```
<div class="row">
  <div class="medium-4 columns">
    <ul class="pricing-table">
      <li class="title">Basic</li>
      ...
    </ul>
  </div>
  <div class="medium-4 columns">
    <ul class="pricing-table">
      <li class="title">Pro</li>
      ...
    </ul>
  </div>
  <div class="medium-4 columns">
    <ul class="pricing-table">
      <li class="title">Premium</li>
      ...
    </ul>
  </div>
</div>
```

Foundation 顶部导航栏

顶部导航栏放在页面头部：



实例

```
<nav class="top-bar" data-topbar>
  <ul class="title-area">
    <li class="name">
      <!-- 如果你不需要标题或图标可以删掉它 -->
      <h1><a href="#">WebSiteName</a></h1>
    </li>
      <!-- 小屏幕上折叠按钮: 去掉 **.menu-icon** 类, 可以去除图标。
      如果需要只显示图片, 可以删除 "Menu" 文本 -->
    <li class="toggle-topbar menu-icon"><a href="#"><span>Menu</spa
  </ul>

  <section class="top-bar-section">
    <ul class="left">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="#">Page 1</a></li>
      <li><a href="#">Page 2</a></li>
      <li><a href="#">Page 3</a></li>
    </ul>
  </section>
</nav>

<!-- 初始化 Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

实例解析

使用 `<nav class="top-bar" data-topbar>` 创建标准工具条。 `.title-area` 类定义了网站logo区域 (必须防止 `li.name` 内) 。屏幕变小后你就可以看到一个 "menu" 按钮。 Foundation 的菜单会根据屏幕尺寸自动折叠喝延展：

小屏幕上，由于尺寸的原因很多选项会被隐藏。 `li.toggle-topbar menu.icon` 类创建了一个菜单的按钮，点击它可以显示被隐藏的选项。 提示: 重置浏览器窗口查看效果。

`.top-bar-section` 定义了导航的链接部分。 `.left` 类指定链接左对齐。 `.active` 类用于显示选中的项，背景为蓝色。

提示: 如果你想导航链接右对齐可以将 `.left` 修改为 `.right` :

实例

```
<section class="top-bar-section">
  <ul class="right">...
```

你可以同时设置左边对齐与右边对齐 :

实例

```
<section class="top-bar-section">
  <ul class="left">
    <li class="active"><a href="#">Home</a></li>
    <li><a href="#">Page 1</a></li>
    <li><a href="#">Page 2</a></li>
  </ul>
  <ul class="right">
    <li><a href="#">Sign Up</a></li>
    <li><a href="#">Login</a></li>
  </ul>
</section>
```

导航栏可以通过 `.divider` 类来添加分割线 (大屏幕上是垂直的线，小屏幕上是水平线):

实例

```
<ul class="left">
  <li class="active"><a href="#">Home</a></li>
  <li class="divider"></li>
  <li><a href="#">Page 1</a></li>
  <li class="divider"></li>
  <li><a href="#">Page 2</a></li>
  <li class="divider"></li>
  <li><a href="#">Page 3</a></li>
  <li class="divider"></li>
</ul>
```

导航栏的下拉菜单

顶部导航栏可以设置下拉菜单。

可以通过在 <li> 元素上添加 .has-dropdown 类来设置下拉菜单:

实例

```
<section class="top-bar-section">
  <ul class="left">
    <li class="active"><a href="#">Home</a></li>
    <li class="has-dropdown">
      <a href="#">Dropdown</a>
      <ul class="dropdown">
        <li><a href="#">First link in dropdown</a></li>
        <li><a href="#">Second link in dropdown</a></li>
        <li class="active"><a href="#">Active link in dropdown</a><
      </ul>
    </li>
  </ul>
</section>
```

分割线

使用 .divider 类来设置下拉菜单的分割线:

实例

```
<ul class="dropdown">
  <li><a href="#">Apple</a></li>
  <li><a href="#">Banana</a></li>
  <li><a href="#">Orange</a></li>
  <li class="divider"></li>
  <li><a href="#">Kale</a></li>
  <li><a href="#">Spinach</a></li>
</ul>
```

下拉菜单标签

在 <li> 内添加 <label> 元素来设置下拉菜单的标签(标题):

实例

```
<ul class="dropdown">
  <li><label>Fruit</label></li>
  <li><a href="#">Apple</a></li>
  <li><a href="#">Banana</a></li>
  <li><a href="#">Orange</a></li>
  <li class="divider"></li>
  <li><label>Vegetable</label></li>
  <li><a href="#">Kale</a></li>
  <li><a href="#">Spinach</a></li>
</ul>
```

内嵌下拉菜单

下拉菜单可以再嵌入一个下拉菜单：

实例

```
<section class="top-bar-section">
  <ul class="left">
    <li class="has-dropdown">
      <a href="#">Dropdown</a>
      <ul class="dropdown">
        <li><label>Level 1</label></li>
        <li><a href="#">Link</a></li>
        <li><a href="#">Link</a></li>
        <li class="has-dropdown">
          <a href="#">New dropdown</a>
          <ul class="dropdown">
            <li><label>Level 2</label></li>
            <li><a href="#">2nd level dropdown</a></li>
            <li><a href="#">2nd level dropdown</a></li>
            <li class="has-dropdown">
              <a href="#">New dropdown</a>
              <ul class="dropdown">
                <li><label>Level 3</label></li>
                <li><a href="#">3rd level dropdown</a></li>
                <li><a href="#">3rd level dropdown</a></li>
              </ul>
            </li>
          </ul>
        </li>
      </ul>
    </li>
  </ul>
</section>
```

可点击

默认情况下导航栏的下拉菜单在鼠标移动过去后显示，我们可以使用 `data-options="is_hover: false"` 属性来设置导航栏在鼠标在点击后显示:

实例

```
<nav class="top-bar" data-topbar data-options="is_hover: false">
```

导航栏上的按钮及图标

你可以在导航栏上放置图标和按钮：

实例

```
<li><a href="#" class="button">Button Link</a></li>
```

你可以在导航栏上放上图标，更多图片样式可以查看 Foundation 图标教程:

实例

```
<head>
<!-- Foundation 图标样式 -->
<link rel="stylesheet" href="http://static.runoob.com/assets/founda
</head>

<ul class="left">
  <li class="active"><a href="#"><i class="fi-home"></i> Home</a></
  <li><a href="#"><i class="fi-torso"></i> Sign Up</a></li>
  <li><a href="#"><i class="fi-magnifying-glass"></i> Search</a></l
</ul>
```

固定导航栏

导航栏可以固定在页面顶部。

页面滚动时导航栏在顶部是不会动的。

要固定导航栏只需要将导航栏放在 `<div class="fixed">` 内即可:

实例

```
<div class="fixed">
  <nav class="top-bar" data-topbar>
    ...
  </nav>
</div>
```

导航栏绝对定位

我们可以将导航栏放在 <div class="sticky"> 内来设置导航栏的绝对定位，当滚动条滚到到该区域时，该导航栏就像固定导航栏一样在顶部不动:

实例

```
<div class="sticky">
  <nav class="top-bar" data-topbar>
    ...
  </nav>
</div>
```

当你使用 .sticky 类时，顶部导航栏在所有屏幕尺寸上将固定不动。如果你需要在指定屏幕上设定只需要在 <nav> 上添加 data-options="sticky_on: small|medium|large" 属性即可：

实例

```
<div class="sticky">
  <!-- 只有在大屏幕上 -->
  <nav class="top-bar" data-topbar data-options="sticky_on: large">
    ..
   </nav>
</div>
```

或者通过数组设置多个屏幕尺寸：

实例

```
<div class="sticky">
  <!-- 小屏幕和大屏幕 (没有中等屏幕)-->
  <nav class="top-bar" data-topbar data-options="sticky_on: [small,
    ..
   </nav>
</div>
```

Foundation 侧边栏

侧边栏导航

Foundation 使用 `<ul class="side-nav">` 创建侧边栏:

实例

```
<ul class="side-nav">
  <li><a href="#">Link 1</a></li>
  <li><a href="#">Link 2</a></li>
  <li><a href="#">Link 3</a></li>
  <li><a href="#">Link 4</a></li>
</ul>
```

激活链接与分割线

已点击的链接可以在 `<li>` 上使用 `.active` 类来标识，使用 `.divider` 类添加水平分割线：

实例

```
<ul class="side-nav">
  <li class="active"><a class="a" href="#">Link 1</a></li>
  <li><a class="a" href="#">Link 2</a></li>
  <li class="divider"></li>
  <li><a class="a" href="#">Link 3</a></li>
  <li><a class="a" href="#">Link 4</a></li>
</ul>
```

网格中的侧边栏

我们可以使用网格设计模式来设置侧边导航栏，实例如下：

实例

```
<div class="row">
  <div class="medium-4 columns" style="background-color:#f1f1f1;">
    <ul class="side-nav">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="#">Learn HTML</a></li>
      ...
    </ul>
  </div>
  <div class="medium-8 columns">
    <h1>Side Nav</h1>
    <p>Some text..</p>
    ...
  </div>
</div>
```

Foundation 滑动导航(Off-Canvas)

侧边栏导航

Off-Canvas 滑动导航现在逐渐在移动页面变得越来越流行了 (点击菜单按钮菜单从左侧滑出):



创建滑动导航

创建滑动导航实例如下：

实例

```
<!-- 最外层div：页面布局 -->
<div class="off-canvas-wrap" data-offcanvas>
  <!-- 内部元素: "工具栏" 内容 (图标, 链接, 描述内容等)-->
  <div class="inner-wrap">
    <nav class="tab-bar">
      <section class="left-small">
        <a class="left-off-canvas-toggle menu-icon" href="#"><span>
      </section>

      <section class="middle tab-bar-section">
        <h1 class="title">Off-canvas Example</h1>
      </section>
    </nav>

    <!-- 滑动菜单 -->
    <aside class="left-off-canvas-menu">
      <!-- Add links or other stuff here -->
      <ul class="off-canvas-list test">
        <li><label>Heading</label></li>
        <li><a href="#">Link 1</a></li>
        <li><a href="#">Link 2</a></li>
        ...
      </ul>
    </aside>

    <!-- 主要内容 -->
    <section class="main-section">
      <h3>Lorem Ipsum</h3>
      <p>....</p>
    </section>

    <!-- 关闭菜单 -->
    <a class="exit-off-canvas"></a>

  </div> <!-- 结束内部内容 -->
</div> <!-- 结束滑动菜单 -->

<!-- 初始化 Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

Foundation 麦哲伦（Magellan）导航



如何创建麦哲伦导航

麦哲伦导航就是一个导航索引，创建方式如下:

实例

```
<div data-magellan-expedition="fixed">
  <dl class="sub-nav">
    <dd data-magellan-arrival="page1"><a href="#page1">Page 1</a></
    <dd data-magellan-arrival="page2"><a href="#page2">Page 2</a></
  </dl>
</div>

<h3 data-magellan-destination="page1">Page1</h3>
<a name="page1"></a>
...

<h3 data-magellan-destination="page2">Page2</h3>
<a name="page2"></a>
...

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

实例解析

在 <div> 元素上添加 `data-magellan-expedition="fixed"` 属性来创建麦哲伦导航。

然后在 `<dd>` 或 `<li>` 上添加 `data-magellan-arrival="_value_"` 属性，后面添加一个与该属性值一样的链接(page1)。

使用 `data-magellan-destination="value"` 属性来控制麦哲伦导航的目标, 后面紧跟的 `<a>` 元素添加 `name="_value_"` 属性。两个属性的值必须与 `data-magellan-arrival` 的值一致 (**page1**)。

最后，初始化 Foundation JS ，用户在滚动页面时导航就会根据当前显示的内容自动切换。

麦哲伦导航头部工具条

麦哲伦导航使用头部工具条实例：

实例

```
<div data-magellan-expedition="fixed">
  <nav class="top-bar" data-topbar>
    ...

    <section class="top-bar-section">
      <ul class="left">
        <li data-magellan-arrival="page1"><a href="#page1">Page 1</
        <li data-magellan-arrival="page2"><a href="#page2">Page 2</
      </ul>
    </section>

  </nav>
</div>

<h3 data-magellan-destination="page1">Page1</h3>
<a name="page1"></a>
...

<h3 data-magellan-destination="page2">Page2</h3>
<a name="page2"></a>
...
```

麦哲伦导航内边距

默认情况下，麦哲伦导航的 `<div>` 元素有 10px 的内边距。可以使用 CSS 移除它：

实例

```
[data-magellan-expedition], [data-magellan-expedition-clone] {
    padding: 0;
}
```

麦哲伦导航选项

使用 data-options 属性修改麦哲伦导航的设置, 例如 `<div data-magellan-expedition="fixed" data-options="destination_t` :

名称	类型	默认	描述	实例
active_class	string	active	指定激活链接的类	
threshold	number	0	指定导航在什么时候需要固定位置。会根据滚动条滚动计算，默认为 0 (auto)。	
destination_threshold	number	20	设该值设定了导航链接显示为激活（蓝色背景）时导航列表距离顶部的值。	
fixed_top	number	0	指定了导航条距离头部的像素值	

Foundation 表单

Foundation 表单控制会自动设置为全局样式:

所有 `<textarea>`, `<select>` 及 `<input>` 元素宽度都为 100%，且带有外边距、内边距、阴影喝鼠标移动效果。

实例

```
<form>
  Input:
  <input type="text" placeholder="Name">

  Textarea:
  <textarea rows="4" placeholder="Address"></textarea>

  Select:
  <select>
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
  </select>
</form>
```

标签

在表单中使用 `<label>` 元素来设置标签，标签可以添加 for 属性和 id 属性。用户在点击标签或输入域时获取输入框焦点：

实例

```
<form>
  <label for="name">Input
    <input type="text" placeholder="Name" id="name">
  </label>

  <label for="adr">Label
    <textarea rows="4" placeholder="Address" id="adr"></textarea>
  </label>

  <label for="num">Select
    <select id="num">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
    </select>
  </label>
</form>
```

如果需要设置标签右对齐，可以使用 `.right` 类：

实例

```
<label class="right">
```

Fieldset

Foundation 渲染 `<fieldset>` 元素的样式如下：

实例

```
<form>
  <fieldset>
    <legend>Fieldset Legend</legend>
    <label>Name
      <input type="text" placeholder="First Name..">
    </label>
    <label>Email
      <input type="text" placeholder="Enter email..">
    </label>
  </fieldset>
</form>
```

错误状态

使用 `.error` 类来设置错误的标签、输入框、文本框样式:

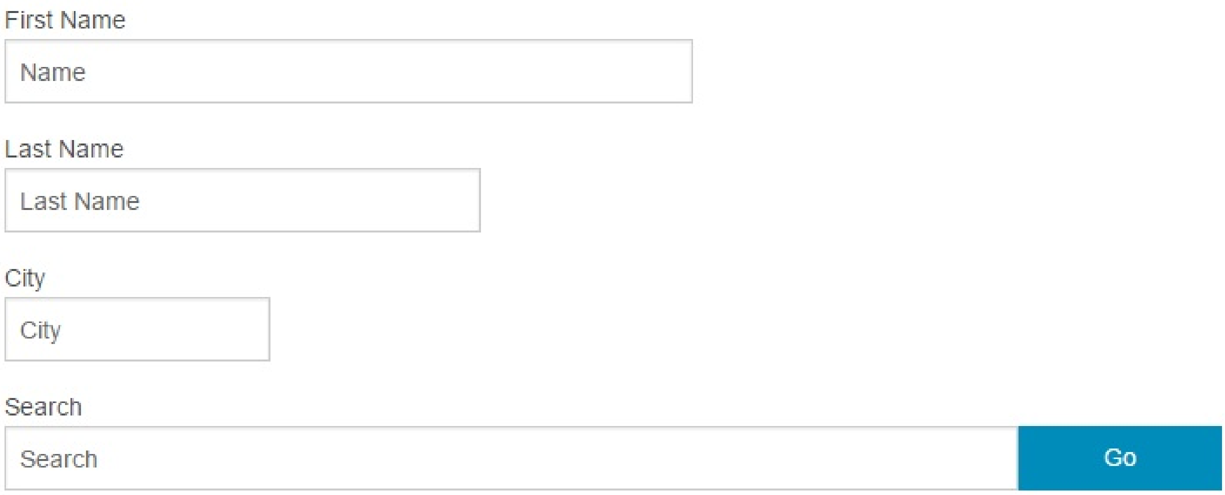
实例

```
<form>
  <label class="error">Error
    <input type="text" placeholder="Name..">
  </label>
  <small class="error">Wrong input</small>

  <textarea rows="4" placeholder="Address"></textarea>
  <small class="error">Wrong input</small>
</form>
```

> 你需要使用 JavaScript 来更新用户输入的错误状态。

Foundation 输入框尺寸



使用网格的列来设置输入框的大小，如 `.large-6` ，`.medium-6` , 等。

更多网格系统知识，可以点击 网格系统 教程。

实例

```
<form>
  <div class="row">
    <div class="large-10 medium-7 columns">
      <label>large-10 medium-7 (100% on small)
        <input type="text" placeholder="Name">
      </label>
    </div>
  </div>

  <div class="row">
    <div class="small-5 columns">
      <label>small-5
        <input type="text" placeholder="Name">
      </label>
    </div>
  </div>

  <div class="row">
    <div class="medium-3 columns">
      <label>medium-3 (100% on small)
        <input type="text" placeholder="Name">
      </label>
    </div>
  </div>
</form>
```

相等大小列

以下演示了相等大小列的实例:

实例

```
<form>
  <div class="row">
    <div class="medium-4 columns">
      <label>medium-4 (100% on small, stacked)
        <input type="text" placeholder="Name">
      </label>
    </div>

    <div class="medium-4 columns">
      <label>medium-4 (100% on small, stacked)
        <input type="text" placeholder="Name">
      </label>
    </div>

    <div class="medium-4 columns">
      <label>medium-4 (100% on small, stacked)
        <input type="text" placeholder="Name">
      </label>
    </div>
  </div>
</form>
```

内联标签

如果你希望你的标签内容显示在左边（不是上边），可以将标签元素 label 放在输入框左边的不同的列上，并使用 `.inline` 类来设置垂直居中:

实例

```
<form>
  <div class="row">
    <div class="small-8">
      <div class="row">
        <div class="small-3 columns">
          <label for="name" class="inline right">Name</label>
        </div>
        <div class="small-9 columns">
          <input type="text" id="name" placeholder="First Name..">
        </div>
      </div>
    </div>
  </div>
</form>
```

前置和后置标签

你可以在 `<div class="row collapse">` 中添加前置和后置标签，元素为： `<_element_ class="postfix">` 或 `<_element_ class="prefix">` 。可以使用网格系统来设置前置和后置标签的大小：

实例

```
<form>
  <div class="row">
    <div class="large-6 columns">
      <div class="row collapse prefix-radius">
        <div class="small-3 columns">
          <span class="prefix">Prefix</span>
        </div>
        <div class="small-9 columns">
          <input type="text" placeholder="Value">
        </div>
      </div>
    </div>
    <div class="large-6 columns">
      <div class="row collapse postfix-radius">
        <div class="small-9 columns">
          <input type="text" placeholder="Value">
        </div>
        <div class="small-3 columns">
          <span class="postfix">Postfix</span>
        </div>
      </div>
    </div>
  </div>
</form>
```

前置和后置标签按钮

可以使用 `<a>` 元素添加 `.button` 类来设置前置和后置按钮:

实例

```
<a href="#" class="postfix button">Go</a>
```

前置和后置标签圆角按钮

实例

```
<form>
  <div class="row">
    <div class="large-6 columns">
      <div class="row collapse prefix-radius">
        <div class="small-3 columns">
          <span class="prefix">Prefix</span>
        </div>
        <div class="small-9 columns">
          <input type="text" placeholder="Value">
        </div>
      </div>
    </div>
    <div class="large-6 columns">
      <div class="row collapse postfix-radius">
        <div class="small-9 columns">
          <input type="text" placeholder="Value">
        </div>
        <div class="small-3 columns">
          <span class="postfix">Postfix</span>
        </div>
      </div>
    </div>
  </div>
  <div class="row">
    <div class="large-6 columns">
      <div class="row collapse prefix-round">
        <div class="small-3 columns">
          <a href="#" class="button prefix">Go</a>
        </div>
        <div class="small-9 columns">
          <input type="text" placeholder="Value">
        </div>
      </div>
    </div>
    <div class="large-6 columns">
      <div class="row collapse postfix-round">
        <div class="small-9 columns">
          <input type="text" placeholder="Value">
        </div>
        <div class="small-3 columns">
          <a href="#" class="button postfix">Go</a>
        </div>
      </div>
    </div>
  </div>
</form>
```

Foundation 开关

开关是在鼠标点击(手指敲击)下在 "On" 和 "Off" 状态下切换：



切换开关使用 `<div class="switch">` 创建。 `<div>` 内添加带有唯一 id 的 `<input type="checkbox">` ， `<label>` 元素的 for 属性需要与 `<input type="checkbox">` 的 id 相匹配：

实例

```
<div class="switch">
  <input id="mySwitch" type="checkbox">
  <label for="mySwitch"></label>
</div>
```

开关大小

使用 `.large` , `.small` 或 `.tiny` 类来设置开关大小：

实例

```
<div class="switch large">...</div>
<div class="switch">...</div>
<div class="switch small">...</div>
<div class="switch tiny">...</div>
```

圆角切换开关

使用 `.radius` 和 `.round` 类来设置圆角切换开关：

实例

```
<div class="switch">...</div>
<div class="switch radius">...</div>
<div class="switch round">...</div>
```

开关组

可以通过设置单选按钮(radio)来设置单个选项。注意: 注意各个选项的 `name` 属性必须一致 (实例中为 "myGroup")。

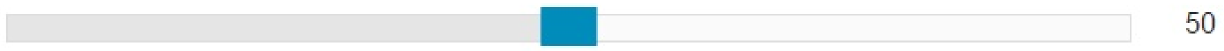
实例

```
<div class="switch">
  <input id="mySwitch1" type="radio" name="myGroup">
  <label for="mySwitch1"></label>
</div>

<div class="switch">
  <input id="mySwitch2" type="radio" name="myGroup" checked>
  <label for="mySwitch2"></label>
</div>
```

Foundation 滑块

Foundation 滑块允许用户通过拖动来选取区间值:



滑块可以通过使用 `<div class="range-slider" data-slider>` 创建。在 `<div>` 内, 添加两个 `<span>` 元素: `<span class="range-slider-handle">` 创建矩形滑块（蓝色背景）, `<span class="range-slider-active-segment">` 是在滑块后的灰色横条，是滑块拖动区域。

注意: 滑块需要使用 JavaScript。所以你需要初始化 oundation JS:

实例

```
<div class="range-slider" data-slider>
  <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

圆角和禁用滑块

使用 `.radius` 和 `.round` 类来添加圆角滑块。使用 `.disabled` 类来禁用滑块：

实例

```
<div class="range-slider" data-slider>..</div>
<div class="range-slider radius" data-slider>...</div>
<div class="range-slider round" data-slider>...</div>
<div class="range-slider disabled" data-slider>...</div>
```

垂直滑块

使用 `.vertical-range` 类和 `data-options="vertical: true;"` 来创建垂直滑块:

实例

```
<div class="range-slider vertical-range" data-slider data-options='
  <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>
```

滑块值

默认情况下，滑块放在横条的中间 (数值为 "50")。可以通过添加 `data-options="initial: _num_"` 属性来修改默认值:

实例

```
<div class="range-slider" data-slider data-options="initial: 80;">
  <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>
```

显示滑块值

如果我们需要在滑块拖动时实时显示值，可以通过在 `<div>` 中添加 `data-options="display_selector:#_id_"` 属性且元素 id 值与滑块的 id 匹配，如下实例：

实例

```
<!-- Display the slider value in this span -->
<span id="mySlider"></span>

<div class="range-slider" data-slider data-options="display_selecto
  <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>
```

组合数据选项

以下实例使用了 `display_selector` 和 `initial` 数据选项:

实例

```
<span id="mySlider"></span>
<div class="range-slider" data-slider data-options="display_selecto
 <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>
```

步长

默认情况下，滑块滑动的增加减少的数量为 "1"。可以通过添加 `data-options="step: _num_;"` 属性来修改步长值。实例中设置为 25:

实例

```
<span id="mySlider"></span>
<div class="range-slider" data-slider data-options="display_selecto
  <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>
```

自定义区间

默认情况下，区间值在 "0" 到 "100"。可以通过添加 data-options `start` 和 `end` 来设置区间值。以下实例设置区间值为 "1" 到 "20":

实例

```
<span id="mySlider"></span>
<div class="range-slider" data-slider data-options="display_selecto
  <span class="range-slider-handle"></span>
  <span class="range-slider-active-segment"></span>
</div>
```

使用网格

以下使用为在网格中使用滑块：

实例

```
<div class="row">
  <div class="small-10 columns">
    <div class="range-slider" data-slider data-options="display_sel
      <span class="range-slider-handle"></span>
      <span class="range-slider-active-segment"></span>
    </div>
  </div>
  <div class="small-2 columns">
    <!-- The display element (Tip: use CSS to perfectly position it
    <span id="mySlider" style="display:block;margin-top:14px;"></sp
  </div>
</div>
```

使用 Input

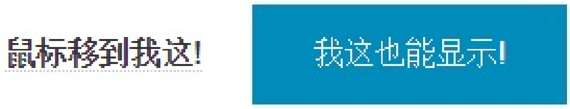
以下实例使用 `<input>` 取代 `<span>` 来显示滑块的值:

实例

```
<div class="row">
  <div class="small-10 columns">
    <div class="range-slider" data-slider data-options="display_sel
      <span class="range-slider-handle"></span>
      <span class="range-slider-active-segment"></span>
    </div>
  </div>
  <div class="small-2 columns">
    <!-- The display element (Tip: use CSS to perfectly position it
    <input type="number" id="mySlider" style="margin-top:7px;" valu
  </div>
</div>
```

Foundation 提示框

提示框在鼠标移动到元素上后显示：



我们可以在任何元素上添加 `data-tooltip` 属性来创建提示信息。使用 `title` 属性来设置提示信息的文本。

注意: 滑块需要使用 JavaScript。所以你需要初始化 oundation JS:

实例

```
<span data-tooltip title="Hooray!">Hover over me!</span>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

`.has-tip` 类可以加粗鼠标移动的文本:

实例

```
<span data-tooltip class="has-tip" title="Hooray!">Hover over me!</
```

提示框显示位置

默认情况下，提示框会出现在元素的底部。

可以使用 `.tip-top` , `.tip-left` , `.tip-right` or `.tip-bottom` (默认) 来设置提示框的位置。

注意: 在小尺寸的屏幕上，提示框的宽带是 100%。

实例

```
<span data-tooltip class="has-tip tip-top" title="Hooray!">Top</spa
<span data-tooltip class="has-tip tip-bottom" title="Hooray!">Botto
<span data-tooltip class="has-tip tip-left" title="Hooray!">Left</s
<span data-tooltip class="has-tip tip-right" title="Hooray!">Right<
```

圆角提示框

`.radius` 和 `.round` 类用于设置圆角提示框：

实例

```
<span data-tooltip class="has-tip" title="Hooray!">Default</span>
<span data-tooltip class="has-tip radius" title="Hooray!">Radius</s
<span data-tooltip class="has-tip round" title="Hooray!">Round</spa
```

Foundation 模态框

模态框是一个显示在页面头部的弹窗。

我们可以在容器元素上(如 `<div id="myModal"`)使用唯一 ID，并添加 `.reveal-modal` 类和 `data-reveal` 属性来添加模态框。我们可以在任何元素上使用 `data-reveal-id="_id_"` 属性阿里打开模态框。 *id* 必须与容器 id 一致 (实例为 "myModal")。

如果你希望在点击模态框之外的区域来关闭模态框。你可以在模态框的关闭按钮 `<a>` 标签上添加 `.close-reveal-modal` 类来实现。

注意: 滑块需要使用 JavaScript。所以你需要初始化 oundation JS:

实例

```
<!-- Trigger the Modal -->
<button type="button" class="button" data-reveal-id="myModal">Click

<!-- The Modal Content -->
<div id="myModal" class="reveal-modal" data-reveal>
  <h2>Heading in Modal.</h2>
  <p>Some text in the modal.</p>
  <p>Some text in the modal.</p>
  <a class="close-reveal-modal">&times;</a>
</div>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

模态框大小

可以在模态框容器上添加以下类来设置模态框的大小：

- `.tiny` : 30% 宽度
- `.small` : 40% 宽度
- `.medium` : 60% 宽度
- `.large` : 70% 宽度
- `.xlarge` : 95% 宽度
- `.full` : 100% 宽度和高度

注意: 在平板、笔记本、PC 电脑上默认为 80% 宽度，在小屏幕设备上是 100% 宽度。

实例

```
<div id="myModal" class="reveal-modal tiny|small|medium|large|xlarg
```

内嵌模态框

你可以在模态框内嵌入模态框，可以在第一个模态框上添加新的触发按钮。你必须为内嵌模态框设置一个唯一的 id：

实例

```
<!-- Trigger the modal -->
<a href="#" class="button" data-reveal-id="myModal">Click To Open M

<!-- The First Modal -->
<div id="myModal" class="reveal-modal" data-reveal>
  <h2>First Modal</h2>
  <p>Some text..</p>
  <p><a href="#" data-reveal-id="secondModal" class="button">Open S
  <a class="close-reveal-modal">&times;</a>
</div>

<!-- The Second Modal -->
<div id="secondModal" class="reveal-modal" data-reveal>
  <h2>Tada! Second Modal</h2>
  <p>Some text..</p>
  <a class="close-reveal-modal">&times;</a>
</div>
```

第二个模态框会取代第一个模态框。如果你希望在打开第二个模态框时，不关闭第一个模态框。可以在第二个模态框上添加 `data-options="multiple_opened:true;"` 属性：

实例

```
<div id="secondModal" class="reveal-modal" data-reveal data-options
```

Foundation Joyride

Joyride 是一个功能向导的 JavaScript 效果，创建实例如下：

实例

```
<!-- Elements that control the tour stops -->
<h3 id="first">First stop!</h3>
<h3 id="second">Second stop!</h3>

<!-- The joyride: must be placed at the bottom of your page, but in
<ol class="joyride-list" data-joyride>
  <li data-id="first">
    <p>First stop. The ride has begun!</p>
  </li>
  <li data-id="second">
    <h4>Second Stop</h4>
    <p>Any valid HTML will work inside the Joyride.</p>
  </li>
  <li data-button="End">
    <h4>End Stop</h4>
    <p>The tour is over. You can either go back to the previous sto
  </li>
</ol>

<!-- Start Joyride Upon Initialization -->
<script>
$(document).ready(function() {
    $(document).foundation('joyride', 'start');
})
</script>
```

实例解析

以上实例中，我们创建了两个元素，每个元素都有独立的 ID。 两个元素设置了 joyride 开始和结束的位置。

我们在 `<ol>` 或 `<ul>` 元素上添加 `data-joyride` 属性和 `.joyride-list` 类来创建 joyride。你需要在文档头部定义它 (在 `<body>` 内的头部)。在每个列表上使用 `<li>` 元素，每个元素添加 `data-id="_value_"` 属性。属性的 *value* 必须与之前元素的 id 相同。所以第一个功能导航 `<h3>` 元素使用 id="first" 必须与 <li> 元素的 data-id="first" 值一致。

如果你没有管理停止的 id，将显示一个模态框。

最后，Joyride 需要使用 JavaScript 初始化它，代码为：

```
$(document).foundation('joyride', 'start');
```

Foundation 均衡器(Equalizer)

我们可以在容器元素添加 `data-equalizer` 属性，并为每个子元素添加 `data-equalizer-watch` 属性来创建一个相同高度的均衡器。最高的元素决定了其他元素的高度。

注意: 滑块需要使用 JavaScript。所以你需要初始化 oundation JS:

实例

```
<div class="row" data-equalizer>
  <div class="medium-4 columns panel" data-equalizer-watch>
    Lorem ipsum...
  </div>
  <div class="medium-4 columns panel" data-equalizer-watch>
    Sed ut...
  </div>
  <div class="medium-4 columns panel" data-equalizer-watch>
    Lorem ipsum...
  </div>
</div>

<!-- Initialize Foundation JS -->
<script>
$(document).ready(function() {
    $(document).foundation();
})
</script>
```

不同屏幕的均衡器

在均衡器上通过添加 `data-equalizer-mq="_value_"` 属性为不同屏幕尺寸设置相同高度。值可以是以下之一：

值	描述	实例
`medium-up`	默认。 创建相同高度的容器， 宽度大于 40.063em	
`large-up`	创建相同高度的容器， 宽度大于 64.063em	
`xlarge-up`	创建相同高度的容器， 宽度大于 90.063em	
`xxlarge-up`	创建相同高度的容器， 宽度大于 120.063em	

嵌套内容

为 `data-equalizer` 和 `data-equalizer-watch` 属性添加相同的值。 这会一起连接到均衡器容器。 重复多次嵌套均衡器，确保他们是匹配的：

实例

```
<!-- The Equalized Container -->
<div class="row" data-equalizer="first">
  <div class="medium-4 columns">
    <div class="panel" data-equalizer-watch="first">
      <h3>Panel</h3>

      <!-- An Equalized Container Inside The Equalized Container --
      <div class="row" data-equalizer="second">
        <div class="panel" data-equalizer-watch="second">
          <h3>Nested Panel</h3>
          <p>Lorem ipsum...</p>
        </div>
        <div class="panel" data-equalizer-watch="second">
          <h3>Nested Panel</h3>
          <p>Lorem ipsum...</p>
        </div>
        <div class="panel" data-equalizer-watch="second">
          <h3>Nested Panel</h3>
          <p>Lorem ipsum...</p>
        </div>
      </div>
      <!-- End Nested Equalized Container -->

    </div>
  </div>
  <div class="medium-4 columns">
    <div class="panel" data-equalizer-watch="first">
      <h3>Panel</h3>
      <p>Ut enim...</p>
    </div>
  </div>
  <div class="medium-4 columns">
    <div class="panel" data-equalizer-watch="first">
      <h3>Panel</h3>
      <p>Lorem ipsum....</p>
    </div>
  </div>
</div>
```

Foundation 网格

Foundation 网格系统

Foundation 网格系统为 12 列。

如果你不需要 12 列，你可以合并一些列，创建一些更大宽度的列。

<table>
<tr><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td></tr>
<tr><td colspan="4">span 4</td><td colspan="4">span 4</td><td colspan="4">span 4</td></tr>
<tr><td colspan="4">span 4</td><td colspan="8">span 8</td></tr>
<tr><td colspan="6">span 6</td><td colspan="6">span 6</td></tr>
<tr><td colspan="12">span 12</td></tr>
</table>

Foundation 的网格系统是响应式的。 列会根据屏幕尺寸自动调整大小。在大尺寸屏幕上，可能是三列，小屏幕尺寸就可能是三个单列，按顺序排列。

网格列

Foundation 网格系统有三个列：

- `.small` (手机端)
- `.medium` (平板设备)
- `.large` (电脑设备：笔记本，台式机)

以上类可以结合使用，创建更灵活的布局

基本的网格结构

以下是基本的 Foundation 网格结构实例:

实例

```
<div class="row">
  <div class="small|medium|large-_num_ columns"></div>
</div>
<div class="row">
  <div class="small|medium|large-_num_ columns"></div>
  <div class="small|medium|large-_num_ columns"></div>
  <div class="small|medium|large-_num_ columns"></div>
</div>
<div class="row">
  ...
</div>
```

首先，创建一行 (`<div class="row">`)。 这是一个水平的垂直列。然后添加列的数量说明 `small-_num_` , `medium-_num_` 及 `large-_num_` 类。注意：列的数量 `_num_` 加起来必须等于 12：

实例

```
<div class="row">
  <div class="small-12 columns">.small-12 yellow</div>
</div>
<div class="row">
  <div class="small-8 columns">.small-8 beige</div>
  <div class="small-4 columns">.small-4 gray</div>
</div>
<div class="row">
  <div class="large-9 small-8 columns">.small-8 .large-9 pink</div>
  <div class="large-3 small-4 columns">.small-4 .large-3 orange</di
</div>
```

实例中，第一行的 <div> 类为 `.small-12` , 这会创建 12 列（100%宽度）。

第二行创建了两个列， `.small-4` 的宽度为33.3%， `.small-8` 的宽度为66.6%。

第三行我们添加了额外的两个列 (`.large-3` 和 `.large-9`)。这意味着如果在大屏幕尺寸下，列就会变为 25% (`.large-3`) 和 75% (`.large-9`)的比例。同时我们也指定了小屏幕上列的比例 33% (`.small-4`) 和 66% (`.small-8`) 。这种组合的方式对于不同屏幕显示效果是非常有帮助的。

网格选项

下表总结了 Foundation 网格系统在多个设备上的说明：

	小型设备	中等设备	大设备
	Phones (<40.0625em (640px))	Tablets (>=40.0625em (640px))	Laptops & Desktops (>=64.0625em (1025px))
网格行为	一直是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的
类前缀	.small-*	.medium-*	.large-*
类的数量	12	12	12
可内嵌	Yes	Yes	Yes
偏移量	Yes	Yes	Yes
列排序	Yes	Yes	Yes

宽屏

网格最大(`.row`) 宽度为 62.5rem。在宽屏上，当宽度大于 62.5rem, 列不会跨越页面的宽度， 即使宽度设定为 100%。但你可以通过 CSS 重新设置 max-width:

实例

```
<style>
.row {
    max-width: 100%;
}
</style>
```

如果你使用默认的 max-width, 但希望背景颜色跨越整个页面宽度，你可以使用 `.row` 包裹整个容器，并指定你需要的背景颜色:

实例

```
<div style="background-color:coral;padding:25px;">
  <div class="row">
    <div class="small-6 columns" style="background-color:yellow;">.
    <div class="small-6 columns" style="background-color:pink;">.sm
  </div>
</div>
```

Foundation 网格 - 水平堆叠

以下实例创建了一个基本的网格系统，在 PC 和平板设备上它是水平平铺的，在手机等小型设备上它是水平堆叠的。

实例

```
<div class="row">
  <div class="medium-6 columns" style="background-color:yellow;">
    <p>菜鸟教程</p>
  </div>
  <div class="medium-6 columns" style="background-color:pink;">
    <p>菜鸟教程</p>
  </div>
</div>
```

提示: `.small|medium|large-*` 类中的数字指定了跨越的列数。所以, `.small-1` 跨越 1 列, `.small-4` 跨越 4 列, `.small-6` 跨越 6 列 (50% 宽度) 等。

注意: 要保证数列加起来是 12 列!

Foundation 网格 - 小型设备

假设我们在小型设备上有一个简单的网格布局，两列，宽度比例为 25% 和 75%。

提示: 小型设备的定义是屏幕小于 `40.0625em` 。

小型设备上我们使用 `.small-*` 类。

```
<div class="small-3 columns">....</div>
<div class="small-9 columns">....</div>
```

以下实例设置了两个列，比例为 25% 和 75% (Foundation 是移动优先: 如果没有特别说明，在大型设备上会继承 .small 类的代码): `.small` in them and use those".

实例

```
<div class="row">
  <div class="small-3 columns" style="background-color:yellow;">
    <p>菜鸟教程</p>
  </div>
  <div class="small-9 columns" style="background-color:pink;">
    <p>菜鸟教程</p>
  </div>
</div>
```

> 注意: 要保证数列加起来是 12 列!

如果需要设置 33.3%/66.6% 的比例，你可以使用 `.small-4` 和 `.small-8` :

实例

```
<div class="row">
  <div class="small-4 columns" style="background-color:yellow;">
    <p>菜鸟教程</p>
  </div>
  <div class="small-8 columns" style="background-color:pink;">
    <p>菜鸟教程</p>
  </div>
</div>
```

Foundation 网格 - 中型设备

上一章节我们介绍了小型设备上我们使用 `.small-*` 类来设置，网格比例为 25%/75%：

```
<div class="small-3 columns">....</div>
<div class="small-9 columns">....</div>
```

在中型设备上我们推荐的比例为 50%/50%。

提示: 中型设备的屏幕尺寸定义在 `40.0625em` 到 `64.0624em` 之间。

中型设备上使用 `.medium-*` 类。

现在我们在中型设备上添加两列：

```
<div class="small-3 medium-6 columns">....</div>
<div class="small-9 medium-6 columns">....</div>
```

以上实例设置了两个列，比例为 25% 和 75% (Foundation 是移动优先: 如果没有特别说明，在大型设备上会继承 .small 类的代码):

小型设备上使用的比例为 25%/75% (`.small-3` 和 `.small-9`)。但在中型设备上使用的比例为 50%/50% (`.medium-6` 和 `.medium-6`) 。

实例

```
<div class="row">
  <div class="small-3 medium-6 columns" style="background-color:yel
    <p>菜鸟教程</p>
  </div>
  <div class="small-9 medium-6 columns" style="background-color:pir
    <p>菜鸟教程</p>
  </div>
</div>
```

注意: 要保证数列加起来是 12 列!

紧在中型设备上使用

以下实例中我们指定了 `.medium-6` 类 (不是 `.small-*`)。这表明在中型或大型设备上比例为 50%/50%。但在小型设备上会水平堆叠 (100% 宽度):

实例

```
<div class="row">
  <div class="medium-6 columns" style="background-color:yellow;">
    <p>菜鸟教程</p>
  </div>
  <div class="medium-6 columns" style="background-color:pink;">
    <p>菜鸟教程</p>
  </div>
</div>
```

Foundation 网格 - 大型设备

上一章节我们介绍了中型设备和小型设备的网格布局，小型设备上使用的比例为 25%/75% (.small-3 和 .small-9)，但在中型设备上使用的比例为 50%/50% (.medium-6 和 .medium-6)：

```
<div class="small-3 medium-6 columns">....</div>
<div class="small-9 medium-6 columns">....</div>
```

在大型设备上我们推荐的比例为 33%/66%。

提示: 大型设备的屏幕尺寸定义大于 64.0625em。

大型设备上使用 `.large-*` 类。

现在我们在大型设备上添加两列：

```
<div class="small-3 medium-6 large-4 columns">....</div>
<div class="small-9 medium-6 large-8 columns">....</div>
```

解析

- 小型设备两个列的比例为 25%/75% (`.small-3` 和 `.small-9`)
- 中型设备两个列的比例为 50%/50% (`.medium-6` 和 `.medium-6`)
- 大型设备两个列的比例为 33%/66% (`.large-4` 和 `.large-8`)

实例

```
<div class="row">
  <div class="small-3 medium-6 large-4 columns" style="background-c
    <p>菜鸟教程</p>
  </div>
  <div class="small-9 medium-6 large-8 columns" style="background-c
    <p>菜鸟教程</p>
  </div>
</div>
```

> 注意: 要保证数列加起来是 12 列!

紧在大型设备上使用

以下实例中我们指定了 `.large-6` 类 (不是 .medium- 和 `*.small-*`)。这表明在大型设备上比例为 50%/50%。但在中型或小型设备上会水平堆叠 (100% 宽度):

实例

```
<div class="row">
  <div class="large-6 columns" style="background-color:yellow;">
    <p>菜鸟教程</p>
  </div>
  <div class="large-6 columns" style="background-color:pink;">
    <p>菜鸟教程</p>
  </div>
</div>
```

Foundation 块状网格

块状网格用于均分页面内容：例如一行内要显示四张图片，不管什么屏幕下都需要均分宽度。

可以使用 `<ul>` 元素加上 `.small|medium|large-block-grid-_num_` 类来创建块状网格。*num* 用于指定均分是数量：

实例

```
<ul class="small-block-grid-3">
  <li><img src="newyork.jpg" alt="New York"></li>
  <li><img src="paris.jpg" alt="Paris"></li>
  <li><img src="sanfran.jpg" alt="San Francisco"></li>
</ul>
```

另一个实例，使用段落：

实例

```
<ul class="small-block-grid-3">
  <li><p>Just a Simple Example Text...</p></li>
  <li><p>Just a Simple Example Text...</p></li>
  <li><p>Just a Simple Example Text...</p></li>
</ul>
```

不同尺寸屏幕显示不同数量

通过添加多个网格块类可以设置不同尺寸屏幕显示不同的块数量:

实例

```
<ul class="small-block-grid-2 medium-block-grid-3 large-block-grid-
  <li><img src="newyork.jpg" alt="New York"></li>
  <li><img src="paris.jpg" alt="Paris"></li>
  <li><img src="sanfran.jpg" alt="San Francisco"></li>
  <li><img src="newyork.jpg" alt="New York"></li>
</ul>
```

Foundation 网格实例

以下我们收集了一些网格常用的实例。

三个均等列

该实例演示了如何创建三个均等列 (33.3%/33.3%/33.3%)，在中型和大型设备上显示三个列，在小型设备上自动堆叠：

实例

```
<div class="row">
  <div class="medium-4 columns" style="background-color:yellow;">
    <p>.medium-4</p>
  </div>
  <div class="medium-4 columns" style="background-color:pink;">
    <p>.medium-4</p>
  </div>
  <div class="medium-4 columns" style="background-color:yellow;">
    <p>.medium-4</p>
  </div>
</div>
```

三个不均等列

该实例演示了如何创建三个不均等列 (25%/50%/25%)，在中型和大型设备上显示三个列，在小型设备上自动堆叠：

实例

```
<div class="row">
  <div class="medium-3 columns" style="background-color:yellow;">
    <p>.medium-3</p>
  </div>
  <div class="medium-6 columns" style="background-color:pink;">
    <p>.medium-6</p>
  </div>
  <div class="medium-3 columns" style="background-color:yellow;">
    <p>.medium-3</p>
  </div>
</div>
```

两个均等列

该实例演示了如何创建两个均等列 (50%/50%)，在小型、中型和大型设备上列的比例始终为 50%/50%：

实例

```
<div class="row">
  <div class="small-6 columns" style="background-color:yellow;">
    <p>.small-6</p>
  </div>
  <div class="small-6 columns" style="background-color:pink;">
    <p>.small-6</p>
  </div>
</div>
```

两个不均等列

该实例演示了如何创建两个不均等列 (33.3%/66.6%)，在小型、中型和大型设备上列的比例始终为 33.3%/66.6%：

实例

```
<div class="row">
  <div class="small-8 columns" style="background-color:yellow;">
    <p>.small-8</p>
  </div>
  <div class="small-4 columns" style="background-color:pink;">
    <p>.small-4</p>
  </div>
</div>
```

修改列的顺序

通过使用 `.small|medium|large-push-*` 和 `.small|medium|large-pull-*` 类来修改列的顺序:

实例

```
<div class="row">
  <div class="small-4 small-8-push columns" style="background-colo
    <p>.small-4 .small-8-push</p>
  </div>
  <div class="small-8 small-4-pull columns" style="background-colo
    <p>.small-8 .small-4-pull</p>
  </div>
</div>
```

嵌套列

你可以使用嵌套网格(列中插入列):

实例

```
<div class="row">
  <div class="small-8 columns">.small-8
    <div class="row">
      <div class="small-8 columns">.small-8 Nested
        <div class="row">
          <div class="small-8 columns">.small-8 Nested Again</div>
          <div class="small-4 columns">.small-4</div>
        </div>
      </div>
      <div class="small-4 columns">.small-4</div>
    </div>
  </div>
  <div class="small-4 columns">.small-4</div>
</div>
```

混合：手机、桌面设备

Foundation 网格系统有三个列: `.small-*` (手机), `.medium-*` (平板), 和 `.large-*` (桌面设备)。这些类可以动态组合使用，让布局更加灵活：

提示： 每个类都能放大，如果你希望小型和大型屏幕设备的宽度一样可以设置指定 `.small-*` 。

实例

```
<div class="row">
  <div class="small-6 large-8 columns">.small-6 .large-8</div>
  <div class="small-6 large-4 columns">.small-6 .large-4</div>
</div>
<div class="row">
  <div class="small-2 large-4 columns">.small-2 .large-2</div>
  <div class="small-4 large-4 columns">.small-4 .large-2</div>
  <div class="small-6 large-4 columns">.small-6 .large-2</div>
</div>
<div class="row">
  <div class="small-3 large-5 columns">.small-3 .large-5</div>
  <div class="small-9 large-7 columns">.small-9 .large-7</div>
</div>
```

混合：手机、平板和桌面设备

实例

```
<div class="row">
  <div class="medium-6 large-8 columns">.medium-6 .large-8</div>
  <div class="medium-6 large-4 columns">.medium-6 .large-4</div>
</div>
<div class="row">
  <div class="small-4 medium-3 large-7 columns">.small-4 .medium-3
  <div class="small-4 medium-6 large-3 columns">.small-4 .medium-6
  <div class="small-4 medium-3 large-2 columns">.small-4 .medium-3
</div>
```

居中列

列居中可以使用 `.small-centered` 类。中型和大型设备可以继承小型设备的居中，但你需要在大型设备上设置居中类 `.large-centered` 。

实例

```
<div class="row">
  <div class="small-4 small-centered columns">small-4 small-centere
</div>
<div class="row">
  <div class="small-6 small-centered columns">small-6 small-centere
</div>
<div class="row">
  <div class="small-6 large-centered columns">small-6 large-centere
</div>
<div class="row">
  <div class="small-8 small-centered large-uncentered columns">smal
</div>
<div class="row">
  <div class="small-10 small-centered columns">small-10 small-cente
</div>
```

列偏移量

可以使用 `.large-offset-*` (或 `.small-offset-*`) 类设置列向右移。 左侧外边距的列数量使用 * 号控制:

实例

```
<div class="row">
  <div class="large-1 columns">1</div>
  <div class="large-11 columns">11</div>
</div>
<div class="row">
  <div class="large-1 columns">1</div>
  <div class="large-10 large-offset-1 columns">10, offset 1</div>
</div>
<div class="row">
  <div class="large-1 columns">1</div>
  <div class="large-9 large-offset-2 columns">9, offset 2</div>
</div>
<div class="row">
  <div class="large-1 columns">1</div>
  <div class="large-8 large-offset-3 columns">8, offset 3</div>
</div>
```

不完整列

如果一行中的列数量之和不是 12 , Foundation 将自动将最后一列向右浮动，并使用空白来填充剩下的列。

可选项 `.end` 类用于设置最后一列的元素向左边浮动:

实例

```
<div class="row">
  <div class="medium-3 columns">.medium-3</div>
  <div class="medium-3 columns">.medium-3</div>
  <div class="medium-3 columns">.medium-3</div>
</div>
<div class="row">
  <div class="medium-3 columns">.medium-3</div>
  <div class="medium-3 columns">.medium-3</div>
  <div class="medium-3 columns end">.medium-3 .end</div>
</div>
```

宽屏

网格 (`.row`) 最大尺寸（ max-width）为 62.5rem。在宽屏设备上尺寸可能大于 62.5rem, 这样列就无法完整填充页面，即便宽度设置为 100%。但是我们可以通过 CSS 来设置新的 max-width:

实例

```
<style>
.row {
    max-width: 100%;
}
</style>
```

如果你想使用默认的 max-width, 但是背景颜色需要跨域整个页面，这时你在容器元素上使用 `.row` 类，并指定你需要的背景颜色:

实例

```
<div style="background-color:coral;padding:25px;">
  <div class="row">
    <div class="small-6 columns" style="background-color:yellow;">.
    <div class="small-6 columns" style="background-color:pink;">.sr
  </div>
</div>
```

Foundation 参考手册

Foundation 图标参考手册

Foundation 图标

Foundation 提供了 283 图标，你可以使用 CSS 来渲染她们：修改大小和颜色。

图标创建语法格式如下：

```
<i class="fi-name"></i>
```

语法中 *name* 部分为图标指定的名称。

要使用图标，你可以将图片的样式文件放置在页面头部 `<head>` 部分:

```
<link rel="stylesheet" href="http://static.runoob.com/assets/foun
```

以上是菜鸟教程使用的图标样式类，大家可以共用。

点我在新页面中查看图标。

General Icons

- heart
- star
- plus
- minus
- x
- check
- upload
- download
- widget
- marker
- refresh
- home
- trash
- paperclip
- lock
- unlock
- calendar
- cloud
- magnifying-glass

- zoom-out
- zoom-in
- wrench
- rss
- share
- flag
- list-thumbnails
- list
- thumbnails
- annotate
- folder
- folder-lock
- folder-add
- clock
- play-video
- crop
- archive
- pencil
- graph-trend
- graph-bar
- graph-horizontal
- graph-pie
- checkbox
- minus-circle
- x-circle
- eye
- database
- results
- results-demographics
- like
- dislike
- upload-cloud
- camera
- alert
- bookmark
- contrast
- mail
- video
- telephone
- comment
- comment-video
- comment-quotes
- comment-minus
- comments
- microphone
- megaphone
- sound
- address-book

- bluetooth
- html5
- css3
- layout
- web
- foundation

Page Icons

- page
- page-csv
- page-doc
- page-pdf
- page-export
- page-export-csv
- page-export-doc
- page-export-pdf
- page-add
- page-remove
- page-delete
- page-edit
- page-search
- page-copy
- page-filled
- page-multiple

Arrow Icons

- arrow-up
- arrow-right
- arrow-down
- arrow-left
- arrows-out
- arrows-in
- arrows-expand
- arrows-compress

People Icons

- torso
- torso-female
- torsos
- torsos-female-male
- torsos-male-female
- torsos-all

- torsos-all-female
- torso-business

Device Icons

- monitor
- laptop
- tablet-portrait
- tablet-landscape
- mobile
- mobile-signal
- usb

Text Editor Icons

- bold
- italic
- underline
- strike
- text-color
- background-color
- superscript
- subscript
- align-left
- align-center
- align-right
- align-justify
- list-numbered
- list-bullet
- indent-more
- indent-less
- print
- save
- photo
- filter
- paint-bucket
- link
- unlink
- quote

Media Control Icons

- play
- stop
- pause

- previous
- rewind
- fast-forward
- next
- record
- play-circle
- volume-none
- volume
- volume-strike
- loop
- shuffle
- eject
- rewind-ten

Ecommerce Icons

- dollar
- euro
- pound
- yen
- bitcoin
- bitcoin-circle
- credit-card
- shopping-cart
- burst
- burst-new
- burst-sale
- paypal
- price-tag
- pricetag-multiple
- shopping-bag
- dollar-bill

Accessibility Icons

- wheelchair
- braille
- closed-caption
- blind
- asl
- hearing-aid
- guide-dog
- universal-access
- telephone-accessible
- elevator
- male

- female
- male-female
- male-symbol
- female-symbol

Social & Brand Icons

- social-500px
- social-adobe
- social-amazon
- social-android
- social-apple
- social-behance
- social-bing
- social-blogger
- social-delicious
- social-designer-news
- social-deviant-art
- social-digg
- social-dribbble
- social-drive
- social-dropbox
- social-evernote
- social-facebook
- social-flickr
- social-forrst
- social-foursquare
- social-game-center
- social-github
- social-google-plus
- social-hacker-news
- social-hi5
- social-instagram
- social-joomla
- social-lastfm
- social-linkedin
- social-medium
- social-myspace
- social-orkut
- social-path
- social-picasa
- social-pinterest
- social-rdio
- social-reddit
- social-skype
- social-skillshare
- social-smashing-mag

- social-snapchat
- social-spotify
- social-squidoo
- social-stack-overflow
- social-steam
- social-stumbleupon
- social-treehouse
- social-tumblr
- social-twitter
- social-vimeo
- social-windows
- social-xbox
- social-yahoo
- social-yelp
- social-youtube
- social-zerply
- social-zurb

Miscellaneous Icons

- compass
- music
- lightbulb
- battery-full
- battery-half
- battery-empty
- projection-screen
- info
- power
- asterisk
- at-sign
- key
- ticket
- book
- book-bookmark
- anchor
- puzzle
- foot
- paw
- mountains
- trees
- sheriff-badge
- first-aid
- trophy
- prohibited
- no-dogs
- no-smoking

- safety-cone
- shield
- crown
- target
- target-two
- die-one
- die-two
- die-three
- die-four
- die-five
- die-six
- skull
- map
- clipboard
- clipboard-pencil
- clipboard-notes

Foundation CSS 参考手册

Foundation 默认设置

Foundation 使用浏览器默认字体大小 (`font-size:100%`)。对于大多数桌面设备的浏览器来说，字体大小默认为 16px。对于移动设备的浏览器，字体默认大小为 12px。 默认的字体为 `"Helvetica Neue"` ， line-height 默认为 `1.5` 。

这些设置是适用于 `<body>` 元素内的元素。

此外， `<p>` 元素与底部的外边距(margin-bottom) 为 1.25rem , line-height 为 1.6。

文本

以下的 HTML 元素，Foundation 设置了独立的样式渲染它，不会采用浏览器默认样式。点击 "尝试一下" 查看在线实例。

元素	描述	在线实例
<h1> - <h6>	h1 - h6 标题	
<a>	浅蓝色的链接，鼠标移动到链接会有下划线	
<small>	浅灰色的副标题文本	
<blockquote>	引用内容模块	
<strong>	加粗文本	
<em>	斜体	
<abbr>	指定单词的缩写，使用该元素文本出现虚线下划线，鼠标移动上去会有提示信息	
<kbd>	接收键盘输入指令: <kbd>CTRL + P</kbd>	
<hr>	水平线	
<code>	代码片段	
	无序列表	
	有序列表	
<dl>	描述性列表	

文本对齐

使用 CSS 类来修改文本的对齐方式：

类	描述	实例
.text-left	左对齐文本	
.text-right	右对齐文本	
.text-center	居中	
.text-justify	两端对齐	

不同尺寸屏幕的对齐

使用 CSS 类来修改文本的不同尺寸屏幕的对齐方式：

类	描述	实例
左对齐		
.small-text-left	所有尺寸屏幕左对齐	
.small-only-text-left	小尺寸屏幕左对齐(宽度小于 40em)	
.medium-text-left	宽度大于 40.0625em 尺寸屏幕左对齐	
.medium-only-text-left	宽度在 40.0625em 到 64em 尺寸的屏幕左对齐	
.large-text-left	宽度大于 64.0625em 尺寸屏幕左对齐	
.large-only-text-left	宽度在 64.0625em 到 90em 尺寸的屏幕左对齐	
.xlarge-text-left	宽度大于 90.0625em 尺寸屏幕左对齐	
.xlarge-only-text-left	宽度在 90.0625em 到 120em 尺寸的屏幕左对齐	
.xxlarge-text-left	宽度大于 120em 尺寸屏幕左对齐	
右对齐		
.small-text-right	所有尺寸屏幕右对齐	
.small-only-text-right	小尺寸屏幕右对齐(宽度小于 40em)	
.medium-text-right	宽度大于 40.0625em 尺寸屏幕右对齐	
.medium-only-text-	宽度在 40.0625em 到 64em 尺寸的屏幕右对	

right	齐
.large-text-right	宽度大于 64.0625em 尺寸屏幕右对齐
.large-only-text-right	宽度在 64.0625em 到 90em 尺寸的屏幕右对齐
.xlarge-text-right	宽度大于 90.0625em 尺寸屏幕右对齐
.xlarge-only-text-right	宽度在 90.0625em 到 120em 尺寸的屏幕右对齐
.xxlarge-text-right	宽度大于 120em 尺寸屏幕右对齐
居中对齐	
.small-text-center	所有尺寸屏幕居中对齐
.small-only-text-center	小尺寸屏幕居中对齐(宽度小于 40em)
.medium-text-center	宽度大于 40.0625em 尺寸屏幕居中对齐
.medium-only-text-center	宽度在 40.0625em 到 64em 尺寸的屏幕居中对齐
.large-text-center	宽度大于 64.0625em 尺寸屏幕居中对齐
.large-only-text-center	宽度在 64.0625em 到 90em 尺寸的屏幕居中对齐
.xlarge-text-center	宽度大于 90.0625em 尺寸屏幕居中对齐
.xlarge-only-text-center	宽度在 90.0625em 到 120em 尺寸的屏幕居中对齐
.xxlarge-text-center	宽度大于 120em 尺寸屏幕居中对齐
两端对齐	
.small-text-justify	所有尺寸屏幕都两端对齐
.small-only-text-justify	小尺寸屏幕两端对齐(宽度小于 40em)
.medium-text-justify	宽度大于 40.0625em 尺寸屏幕两端对齐
.medium-only-text-justify	宽度在 40.0625em 到 64em 尺寸的屏幕两端对齐
.large-text-justify	宽度大于 64.0625em 尺寸屏幕两端对齐
.large-only-text-justify	宽度在 64.0625em 到 90em 尺寸的屏幕两端对齐
.xlarge-text-justify	宽度大于 90.0625em 尺寸屏幕两端对齐
.xlarge-only-text-justify	宽度在 90.0625em 到 120em 尺寸的屏幕两端对齐

.xxlarge-text-justify	宽度大于 120em 尺寸屏幕两端对齐

其他

类	描述	实例
.left	元素向左浮动	
.right	元素向右浮动	
.clearfix	清除浮动 - 必须添加在浮动元素的父元素上	
.hide	隐藏元素 (CSS `display: none`)	
.list-inline	将所有元素设置在同一行	
.lead	让 <p> 元素更突出	
.subheader	设置浅色的 <h1> - <h6> 元素	

Foundation CSS 可见性

根据屏幕尺寸显示元素

以下类会根据设备(屏幕尺寸)来显示元素。

类	描述
.show-for-small-only	只在小型设备上显示元素 (屏幕宽度小于 40.0625em)
.show-for-medium-up	在中型及以上设备上显示元素 (屏幕宽度大于 40.0625em)
.show-for-medium-only	只在中型设备上显示元素 (屏幕宽度在 40.0625em 到 64.0625em 之间)
.show-for-large-up	在大型及以上设备上显示元素 (屏幕宽度大于 64.0625em)
.show-for-large-only	只在大型设备上显示元素 (屏幕宽度在 64.0625em 到 90.0625em 之间)
.show-for-xlarge-up	在更大型及以上设备上显示元素 (屏幕宽度大于 90.0625em)
.show-for-xlarge-only	只在更大型及以上设备上显示元素 (屏幕宽度在 90.0625em 到 120.0625em之间)
.show-for-xxlarge-up	在超大型及以上设备上显示元素 (屏幕宽度大于 120.0625em)

以下实例演示了以上所有 `.show-` 类的可见性。

```
<p class="show-for-small-only">你在小型设备上。</p>
<p class="show-for-medium-up">你在中型、大型、更大型、超大型的设备上。</p
<p class="show-for-medium-only">你在中型设备上。</p>
<p class="show-for-large-up">你在大型、更大型、超大型的设备上</p>
<p class="show-for-large-only">你在大型设备上。</p>
<p class="show-for-xlarge-up">你在更大型、超大型的设备上。</p>
<p class="show-for-xlarge-only">你在更大型设备上。</p>
<p class="show-for-xxlarge-up">你在超大型设备上。</p>
```

根据屏幕尺寸隐藏元素

以下类会根据设备(屏幕尺寸)来隐藏元素。

类	描述
.hide-for-small-only	只在小型设备上隐藏元素 (屏幕宽度小于 40.0625em)
.hide-for-medium-up	在中型及以上设备上隐藏元素 (屏幕宽度大于 40.0625em)
.hide-for-medium-only	只在中型设备上隐藏元素 (屏幕宽度在 40.0625em 到 64.0625em 之间)
.hide-for-large-up	在大型及以上设备上隐藏元素 (屏幕宽度大于 64.0625em)
.hide-for-large-only	只在大型设备上隐藏元素 (屏幕宽度在 64.0625em 到 90.0625em 之间)
.hide-for-xlarge-up	在更大型及以上设备上隐藏元素 (屏幕宽度大于 90.0625em)
.hide-for-xlarge-only	只在更大型及以上设备上隐藏元素 (屏幕宽度在 90.0625em 到 120.0625em之间)
.hide-for-xxlarge-up	在超大型及以上设备上隐藏元素 (屏幕宽度大于 120.0625em)

```
<p class="hide-for-small-only">你不在小型设备上。</p>
<p class="hide-for-medium-up">你不在中型、大型、更大型、超大型的设备上。</
<p class="hide-for-medium-only">你不在中型设备上。</p>
<p class="hide-for-large-up">你不在大型、更大型、超大型的设备上。</p>
<p class="hide-for-large-only">你不在大型设备上。</p>
<p class="hide-for-xlarge-up">你不在更大型、超大型的设备上。</p>
<p class="hide-for-xlarge-only">你不在更大型设备上。</p>
<p class="hide-for-xxlarge-up">你不在超大型设备上。</p>
```

根据屏幕方向显示元素

以下类会根据设备(屏幕尺寸)来隐藏元素。

我们可以设置元素在不同方向是是否显示或隐藏。笔记本等桌面设备一般是横向的，但是手机和平板设备可以是横向或纵向，我们可以根据用户手机拿的方向来设置元素隐藏与显示：

类	描述
.show-for-landscape	在横向时显示元素（纵向隐藏）
.show-for-portrait	在纵向时显示元素（横向隐藏）

下面实例根据使用的方向显示文本内容：

实例

```
<p class="show-for-landscape">文本只在横向显示。</p>
<p class="show-for-portrait">文本只在纵向显示。</p>
```

触屏设备的显示与隐藏

你可以根据设备是否支持触摸来显示与隐藏元素。

类	描述
.show-for-touch	在支持触屏的设备上显示(不支持的设备上隐藏)
.hide-for-touch	在支持触屏的设备上隐藏(不支持的设备上显示)

下面实例根据设备是否支持触摸来显示文本内容：

实例

```
<p class="show-for-touch">你的设备支持触屏。</p>
<p class="hide-for-touch">你的设备不支持触屏。</p>
```

W3School Javascript 教程

来源：

- JavaScript教程
- HTML DOM教程
- Ajax教程

整理：飞龙

JavaScript 基础

JavaScript 简介

JavaScript 是世界上最流行的编程语言。

这门语言可用于 **HTML** 和 **web**，更可广泛用于服务器、**PC**、笔记本电脑、平板电脑和智能手机等设备。

JavaScript 是脚本语言

JavaScript 是一种轻量级的编程语言。

JavaScript 是可插入 HTML 页面的编程代码。

JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。

JavaScript 很容易学习。

您将学到什么

下面是您将在本教程中学到的主要内容。

JavaScript：写入 HTML 输出

实例

```
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph</p>");
```

提示：您只能在 HTML 输出中使用 document.write。如果您在文档加载后使用该方法，会覆盖整个文档。

JavaScript：对事件作出反应

实例

```
<button type="button" onclick="alert('Welcome!')">点击这里</button>
```

alert() 函数在 JavaScript 中并不常用，但它对于代码测试非常方便。

onclick 事件只是您即将在本教程中学到的众多事件之一。

JavaScript：改变 HTML 内容

使用 JavaScript 来处理 HTML 内容是非常强大的功能。

实例

```
x=document.getElementById("demo")  //查找元素
x.innerHTML="Hello JavaScript";    //改变内容
```

您会经常看到 document.getElementByID("*some id*")。这个方法是 HTML DOM 中定义的。

DOM（文档对象模型）是用以访问 HTML 元素的正式 W3C 标准。

您将在本教程的多个章节中学到有关 HTML DOM 的知识。

JavaScript：改变 HTML 图像

本例会动态地改变 HTML <image> 的来源 (src)：

The Light bulb

点击灯泡就可以打开或关闭这盏灯

JavaScript 能够改变任意 HTML 元素的大多数属性，而不仅仅是图片。

JavaScript：改变 HTML 样式

改变 HTML 元素的样式，属于改变 HTML 属性的变种。

实例

```
x=document.getElementById("demo")  //找到元素
x.style.color="#ff0000";           //改变样式
```

JavaScript：验证输入

JavaScript 常用于验证用户的输入。

实例

```
if isNaN(x) {alert("Not Numeric")};
```

您知道吗？

提示：JavaScript 与 Java 是两种完全不同的语言，无论在概念还是设计上。

Java（由 Sun 发明）是更复杂的编程语言。

ECMA-262 是 JavaScript 标准的官方名称。

JavaScript 由 Brendan Eich 发明。它于 1995 年出现在 Netscape 中（该浏览器已停止更新），并于 1997 年被 ECMA（一个标准协会）采纳。

课外阅读

JavaScript 高级教程：JavaScript 历史、 JavaScript 实现

JavaScript 使用

HTML 中的脚本必须位于 **<script>** 与 **</script>** 标签之间。

脚本可被放置在 **HTML** 页面的 **<body>** 和 **<head>** 部分中。

JavaScript 输出

JavaScript 通常用于操作 **HTML** 元素。

操作 **HTML** 元素

如需从 JavaScript 访问某个 HTML 元素，您可以使用 document.getElementById(*id*) 方法。

请使用 "id" 属性来标识 HTML 元素：

例子

通过指定的 id 来访问 HTML 元素，并改变其内容：

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>

<p id="demo">My First Paragraph</p>

<script>
`document.getElementById("demo").innerHTML="My First JavaScript";`
</script>

</body>
</html>
```

JavaScript 由 web 浏览器来执行。在这种情况下，浏览器将访问 id="demo" 的 HTML 元素，并把它的内容（innerHTML）替换为 "My First JavaScript"。

写到文档输出

下面的例子直接把 <p> 元素写到 HTML 文档输出中：

实例

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>

<script>
`document.write("<p>My First JavaScript</p>");`
</script>

</body>
</html>
```

警告

请使用 document.write() 仅仅向文档输出写内容。

如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖：

实例

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>

<p>My First Paragraph.</p>

<button onclick="myFunction()">点击这里</button>

<script>
function myFunction()
{
document.write("糟糕！文档消失了。");
}
</script>

</body>
</html>
```

Windows 8 中的 JavaScript

提示：微软支持通过 JavaScript 创建 Windows 8 app。

对于因特网和视窗操作系统，JavaScript 都意味着未来。

JavaScript 语句

JavaScript 语句

JavaScript 语句向浏览器发出的命令。语句的作用是告诉浏览器该做什么。

下面的 JavaScript 语句向 id="demo" 的 HTML 元素输出文本 "Hello World"：

```
document.getElementById("demo").innerHTML="Hello World";
```

分号；

分号用于分隔 JavaScript 语句。

通常我们在每条可执行的语句结尾添加分号。

使用分号的另一用处是在一行中编写多条语句。

提示：您也可能看到不带有分号的案例。

在 JavaScript 中，用分号来结束语句是可选的。

JavaScript 代码

JavaScript 代码（或者只有 JavaScript）是 JavaScript 语句的序列。

浏览器会按照编写顺序来执行每条语句。

本例将操作两个 HTML 元素：

实例

```
document.getElementById("demo").innerHTML="Hello World";
document.getElementById("myDIV").innerHTML="How are you?";
```

JavaScript 代码块

JavaScript 语句通过代码块的形式进行组合。

块由左花括号开始，由右花括号结束。

块的作用是使语句序列一起执行。

JavaScript 函数是将语句组合在块中的典型例子。

下面的例子将运行可操作两个 HTML 元素的函数：

实例

```
function myFunction()
{
document.getElementById("demo").innerHTML="Hello World";
document.getElementById("myDIV").innerHTML="How are you?";
}
```

您将在稍后的章节学到更多有关函数的知识。

JavaScript 对大小写敏感。

JavaScript 对大小写是敏感的。

当编写 JavaScript 语句时，请留意是否关闭大小写切换键。

函数 getElementById 与 getElementbyID 是不同的。

同样，变量 myVariable 与 MyVariable 也是不同的。

空格

JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。下面的两行代码是等效的：

```
var name="Hello";
var name = "Hello";
```

对代码行进行折行

您可以在文本字符串中使用反斜杠对代码行进行换行。下面的例子会正确地显示：

```
document.write("Hello \
World!");
```

不过，您不能像这样折行：

```
document.write \
("Hello World!");
```

您知道吗？

提示：JavaScript 是脚本语言。浏览器会在读取代码时，逐行地执行脚本代码。而对于传统编程来说，会在执行前对所有代码进行编译。

课外阅读

JavaScript 高级教程： ECMAScript 语法

JavaScript 注释

JavaScript 注释可用于提高代码的可读性。

JavaScript 注释

JavaScript 不会执行注释。

我们可以添加注释来对 JavaScript 进行解释，或者提高代码的可读性。

单行注释以 // 开头。

例子

下面的例子使用单行注释来解释代码：

```
// 输出标题：
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
// 输出段落：
document.getElementById("myP").innerHTML="This is my first paragrap
```

JavaScript 多行注释

多行注释以 / 开始，以 / 结尾。

下面的例子使用多行注释来解释代码：

例子

```
/*
下面的这些代码会输出
一个标题和一个段落
并将代表主页的开始
*/
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
document.getElementById("myP").innerHTML="This is my first paragrap
```

使用注释来阻止执行

例子 1

在下面的例子中，注释用于阻止其中一条代码行的执行（可用于调试）：

```
//document.getElementById("myH1").innerHTML="Welcome to my Homepage
document.getElementById("myP").innerHTML="This is my first paragrap
```

例子 2

在下面的例子中，注释用于阻止代码块的执行（可用于调试）：

```
/*
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
document.getElementById("myP").innerHTML="This is my first paragrap
*/
```

在行末使用注释

在下面的例子中，我们把注释放到代码行的结尾处：

例子

```
var x=5;    // 声明 x 并把 5 赋值给它
var y=x+2;  // 声明 y 并把 x+2 赋值给它
```

课外阅读

JavaScript 高级教程： ECMAScript 语法

JavaScript 变量

变量是存储信息的容器。

实例

```
var x=2;
var y=3;
var z=x+y;
```

就像代数那样

```
x=2
y=3
z=x+y
```

在代数中，我们使用字母（比如 x）来保存值（比如 2）。

通过上面的表达式 z=x+y，我们能够计算出 z 的值为 5。

在 JavaScript 中，这些字母被称为变量。

提示：您可以把变量看做存储数据的容器。

JavaScript 变量

与代数一样，JavaScript 变量可用于存放值（比如 x=2）和表达式（比如 z=x+y）。

变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, totalvolume）。

- 变量必须以字母开头
- 变量也能以 $ 和 _ 符号开头（不过我们不推荐这么做）
- 变量名称对大小写敏感（y 和 Y 是不同的变量）

提示：JavaScript 语句和 JavaScript 变量都对大小写敏感。

JavaScript 数据类型

JavaScript 变量还能保存其他数据类型，比如文本值 (name="Bill Gates")。

在 JavaScript 中，类似 "Bill Gates" 这样一条文本被称为字符串。

JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

例子

```
var pi=3.14;
var name="Bill Gates";
var answer='Yes I am!';
```

声明（创建） JavaScript 变量

在 JavaScript 中创建变量通常称为“声明”变量。

我们使用 var 关键词来声明变量：

```
var carname;
```

变量声明之后，该变量是空的（它没有值）。

如需向变量赋值，请使用等号：

```
carname="Volvo";
```

不过，您也可以在声明变量时对其赋值：

```
var carname="Volvo";
```

例子

在下面的例子中，我们创建了名为 carname 的变量，并向其赋值 "Volvo"，然后把它放入 id="demo" 的 HTML 段落中：

```
<p id="demo"></p>
var carname="Volvo";
document.getElementById("demo").innerHTML=carname;
```

提示：一个好的编程习惯是，在代码开始处，统一对需要的变量进行声明。

一条语句，多个变量

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

```
var name="Gates", age=56, job="CEO";
```

声明也可横跨多行：

```
var name="Gates",
age=56,
job="CEO";
```

Value = undefined

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。

在执行过以下语句后，变量 carname 的值将是 undefined：

```
var carname;
```

重新声明 JavaScript 变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

```
var carname="Volvo";
var carname;
```

JavaScript 算数

您可以通过 JavaScript 变量来做算数，使用的是 = 和 + 这类运算符：

例子

```
y=5;
x=y+2;
```

您将在本教程稍后的章节学到更多有关 JavaScript 运算符的知识。

课外阅读

JavaScript 高级教程： ECMAScript 语法、 ECMAScript 变量

JavaScript 数据类型

字符串、数字、布尔、数组、对象、**Null**、**Undefined**

JavaScript 拥有动态类型

JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

实例

```
var x                // x 为 undefined
var x = 6;           // x 为数字
var x = "Bill";      // x 为字符串
```

JavaScript 字符串

字符串是存储字符（比如 "Bill Gates"）的变量。

字符串可以是引号中的任意文本。您可以使用单引号或双引号：

实例

```
var carname="Bill Gates";
var carname='Bill Gates';
```

您可以在字符串中使用引号，只要不匹配包围字符串的引号即可：

实例

```
var answer="Nice to meet you!";
var answer="He is called 'Bill'";
var answer='He is called "Bill"';
```

您将在本教程的高级部分学到更多关于字符串的知识。

JavaScript 数字

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：

实例

```
var x1=34.00;      //使用小数点来写
var x2=34;         //不使用小数点来写
```

极大或极小的数字可以通过科学（指数）计数法来书写：

实例

```
var y=123e5;      // 12300000
var z=123e-5;     // 0.00123
```

您将在本教程的高级部分学到更多关于数字的知识。

JavaScript 布尔

布尔（逻辑）只能有两个值：true 或 false。

```
var x=true
var y=false
```

布尔常用在条件测试中。您将在本教程稍后的章节中学到更多关于条件测试的知识。

JavaScript 数组

下面的代码创建名为 cars 的数组：

```
var cars=new Array();
cars[0]="Audi";
cars[1]="BMW";
cars[2]="Volvo";
```

或者 (condensed array):

```
var cars=new Array("Audi","BMW","Volvo");
```

或者 (literal array):

实例

```
var cars=["Audi","BMW","Volvo"];
```

数组下标是基于零的，所以第一个项目是 [0]，第二个是 [1]，以此类推。

您将在本教程稍后的章节中学到更多关于数组的知识。

JavaScript 对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

```
var person={firstname:"Bill", lastname:"Gates", id:5566};
```

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。

空格和折行无关紧要。声明可横跨多行：

```
var person={
firstname : "Bill",
lastname  : "Gates",
id        :  5566
};
```

对象属性有两种寻址方式：

实例

```
name=person.lastname;
name=person["lastname"];
```

您将在本教程稍后的章节中学到更多关于对象的知识。

Undefined 和 Null

Undefined 这个值表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量。

实例

```
cars=null;
person=null;
```

声明变量类型

当您声明新变量时，可以使用关键词 "new" 来声明其类型：

```
var carname=new String;
var x=      new Number;
var y=      new Boolean;
var cars=   new Array;
var person= new Object;
```

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

课外阅读

JavaScript 高级教程：

JavaScript 对象

JavaScript 中的所有事物都是对象：字符串、数字、数组、日期，等等。

在 **JavaScript** 中，对象是拥有属性和方法的数据。

属性和方法

属性是与对象相关的值。

方法是能够在对象上执行的动作。

举例：汽车就是现实生活中的对象。

汽车的属性：

```
car.name=Fiat

car.model=500

car.weight=850kg

car.color=white
```

汽车的方法：

```
car.start()

car.drive()

car.brake()
```

汽车的属性包括名称、型号、重量、颜色等。

所有汽车都有这些属性，但是每款车的属性都不尽相同。

汽车的方法可以是启动、驾驶、刹车等。

所有汽车都拥有这些方法，但是它们被执行的时间都不尽相同。

JavaScript 中的对象

在 JavaScript 中，对象是数据（变量），拥有属性和方法。

当您像这样声明一个 JavaScript 变量时：

```
var txt = "Hello";
```

您实际上已经创建了一个 JavaScript 字符串对象。字符串对象拥有内建的属性 length。对于上面的字符串来说，length 的值是 5。字符串对象同时拥有若干个内建的方法。

属性：

```
txt.length=5
```

方法：

```
txt.indexOf()

txt.replace()

txt.search()
```

提示：在面向对象的语言中，属性和方法常被称为对象的成员。

在本教程稍后的章节中，您将学到有关字符串对象的更多属性和方法。

创建 JavaScript 对象

JavaScript 中的几乎所有事务都是对象：字符串、数字、数组、日期、函数，等等。

你也可以创建自己的对象。

本例创建名为 "person" 的对象，并为其添加了四个属性：

实例

```
person=new Object();
person.firstname="Bill";
person.lastname="Gates";
person.age=56;
person.eyecolor="blue";
```

创建新 JavaScript 对象有很多不同的方法，并且您还可以向已存在的对象添加属性和方法。

您将在本教程稍后的章节学到更多相关的内容。

访问对象的属性

访问对象属性的语法是：

```
_objectName_._propertyName_
```

本例使用 String 对象的 length 属性来查找字符串的长度：

```
var message="Hello World!";
var x=message.length;
```

在以上代码执行后，x 的值是：

```
12
```

访问对象的方法

您可以通过下面的语法调用方法：

```
_objectName_._methodName_()
```

这个例子使用 String 对象的 toUpperCase() 方法来把文本转换为大写：

```
var message="Hello world!";
var x=message.toUpperCase();
```

在以上代码执行后，x 的值是：

```
HELLO WORLD!
```

您知道吗？

提示：在面向对象的语言中，使用 camel-case 标记法的函数是很常见的。您会经常看到 someMethod() 这样的函数名，而不是 some_method()。

课外书

如需更多有关 *JavaScript* 对象的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 面向对象技术

本节简要介绍了面向对象技术的术语、面向对象语言的要求以及对象的构成。

ECMAScript 对象应用

本节讲解了如何声明和实例化对象，如何引用和废除对象，以及绑定的概念。

ECMAScript 对象类型

本节介绍了 ECMAScript 的三种类型：本地对象、内置对象和宿主对象，并提供了指向相关参考手册的链接。

ECMAScript 对象作用域

本节讲解了 ECMAScript 作用域以及 this 关键字。

ECMAScript 定义类或对象

本节详细讲解了创建 ECMAScript 对象或类的各种方式。

ECMAScript 修改对象

本节讲解了如何通过创建新方法或重定义已有方法来修改对象。

JavaScript 函数

函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

实例

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>

<body>
<button onclick="myFunction()">点击这里</button>
</body>
</html>
```

JavaScript 函数语法

函数就是包裹在花括号中的代码块，前面使用了关键词 function：

```
function functionname()
{
这里是要执行的代码
}
```

当调用该函数时，会执行函数内的代码。

可以在某事件发生时直接调用函数（比如当用户点击按钮时），并且可由 JavaScript 在任何位置进行调用。

提示：JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

调用带参数的函数

在调用函数时，您可以向其传递值，这些值被称为参数。

这些参数可以在函数中使用。

您可以发送任意多的参数，由逗号 (,) 分隔：

```
myFunction(_argument1_,_argument2_)
```

当您声明函数时，请把参数作为变量来声明：

```
function myFunction(`var1`,`var2`)
{
这里是要执行的代码
}
```

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递的参数的给定的值，以此类推。

实例

```
<button onclick="myFunction('Bill Gates','CEO')">点击这里</button>

<script>
function myFunction(name,job)
{
alert("Welcome " + `name` + ", the " + `job`);
}
</script>
```

上面的函数会当按钮被点击时提示 "Welcome Bill Gates, the CEO"。

函数很灵活，您可以使用不同的参数来调用该函数，这样就会给出不同的消息：

实例

```
<button onclick="myFunction('Harry Potter','Wizard')">点击这里</butt
<button onclick="myFunction('Bob','Builder')">点击这里</button>
```

根据您点击的不同的按钮，上面的例子会提示 "Welcome Harry Potter, the Wizard" 或 "Welcome Bob, the Builder"。

带有返回值的函数

有时，我们会希望函数将值返回调用它的地方。

通过使用 return 语句就可以实现。

在使用 return 语句时，函数会停止执行，并返回指定的值。

语法

```
function myFunction()
{
var x=5;
`return x;`
}
```

上面的函数会返回值 5。

注释：整个 JavaScript 并不会停止执行，仅仅是函数。JavaScript 将继续执行代码，从调用函数的地方。

函数调用将被返回值取代：

```
var myVar=myFunction();
```

myVar 变量的值是 5，也就是函数 "myFunction()" 所返回的值。

即使不把它保存为变量，您也可以使用返回值：

```
document.getElementById("demo").innerHTML=myFunction();
```

"demo" 元素的 innerHTML 将成为 5，也就是函数 "myFunction()" 所返回的值。

您可以使返回值基于传递到函数中的参数：

实例

计算两个数字的乘积，并返回结果：

```
function myFunction(a,b)
{
return a*b;
}

document.getElementById("demo").innerHTML=myFunction(4,3);
```

"demo" 元素的 innerHTML 将是：

```
12
```

在您仅仅希望退出函数时，也可使用 return 语句。返回值是可选的：

```
function myFunction(a,b)
{
if (a>b)
  {
  return;
  }
x=a+b
}
```

如果 a 大于 b，则上面的代码将退出函数，并不会计算 a 和 b 的总和。

局部 JavaScript 变量

在 JavaScript 函数内部声明的变量（使用 var）是局部变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。

您可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。

只要函数运行完毕，本地变量就会被删除。

全局 JavaScript 变量

在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问它。

JavaScript 变量的生存期

JavaScript 变量的生命期从它们被声明的时间开始。

局部变量会在函数运行以后被删除。

全局变量会在页面关闭后被删除。

向未声明的 JavaScript 变量来分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

这条语句：

```
carname="Volvo";
```

将声明一个全局变量 carname，即使它在函数内执行。

课外书

如需更多有关 *JavaScript* 函数的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 函数概述

本节讲解函数的概念，ECMAScript 如何声明并调用函数，以及函数如何返回值。

ECMAScript arguments 对象

本节介绍了此对象的基本用法，然后讲解了如何使用 length 属性来测定函数的参数数目，以及模拟函数重载。

ECMAScript Function 对象（类）

本节讲解了如何使用 Function 类创建函数，然后介绍了 Function 对象的属性和方法。

ECMAScript 闭包（closure）

本节讲解了闭包（closure）的概念，并分别为您展示了简单和稍复杂的两个闭包实例。

JavaScript 运算符

运算符 **=** 用于赋值。

运算符 **+** 用于加值。

运算符 = 用于给 JavaScript 变量赋值。

算术运算符 + 用于把值加起来。

```
y=5;
z=2;
x=y+z;
```

在以上语句执行后，x 的值是 7。

JavaScript 算术运算符

算术运算符用于执行变量与/或值之间的算术运算。

给定 *y=5*，下面的表格解释了这些算术运算符：

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数 (保留整数)	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

JavaScript 赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

给定 *x=10* 和 *y=5*，下面的表格解释了赋值运算符：

运算符	例子	等价于	结果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

用于字符串的 + 运算符

- 运算符用于把文本值或字符串变量加起来（连接起来）。

如需把两个或多个字符串变量连接起来，请使用 + 运算符。

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

在以上语句执行后，变量 txt3 包含的值是 "What a verynice day"。

要想在两个字符串之间增加空格，需要把空格插入一个字符串之中：

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

或者把空格插入表达式中：

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

在以上语句执行后，变量 txt3 包含的值是：

"What a very nice day"

对字符串和数字进行加法运算

请看这些例子：

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

TIY

规则是：

如果把数字与字符串相加，结果将成为字符串。

课外书

如需更多有关 *JavaScript* 运算符的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 一元运算符

一元运算符只有一个参数，即要操作的对象或值。本节讲解 ECMAScript 中最简单的运算符 - 一元运算符。

ECMAScript 位运算符

位运算符是在数字底层进行操作的。本节深入讲解了有关整数的知识，并介绍了 ECMAScript 的各种位运算符。

ECMAScript Boolean 运算符

Boolean 运算符非常重要。本节深入讲解三种 Boolean 运算符：NOT、AND 和 OR。

ECMAScript 乘性运算符

本节讲解 ECMAScript 的乘性运算符：乘法、除法、取模运算符，以及它们的特殊行为。

ECMAScript 加性运算符

本节讲解 ECMAScript 的加性运算符：加法、减法运算符，以及它们的特殊行为。

ECMAScript 关系运算符

关系运算符执行的是比较运算。本节讲解关系运算符的常规比较方式，以及如何比较字符串与数字。

ECMAScript 等性运算符

等性运算符用于判断变量是否相等。ECMAScript 提供两套等性运算符：等号和非等号，以及全等号和非全等号。

ECMAScript 条件运算符

本节讲解 ECMAScript 中的条件运算符。

ECMAScript 赋值运算符

本节讲解 ECMAScript 中的赋值运算符。

ECMAScript 逗号运算符符

本节讲解 ECMAScript 中的逗号运算符。

JavaScript 比较和逻辑运算符

比较和逻辑运算符用于测试 **true** 或 **false**。

比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

给定 x=5，下面的表格解释了比较运算符：

运算符	描述	例子
==	等于	x==8 为 false
===	全等（值和类型）	x===5 为 true；x==="5" 为 false
!=	不等于	x!=8 为 true
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

如何使用

可以在条件语句中使用比较运算符对值进行比较，然后根据结果来采取行动：

```
if (age<18) document.write("Too young");
```

您将在本教程的下一节中学习更多有关条件语句的知识。

逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
&124;&124;	or	(x==5 &124;&124; y==5) 为 false
!	not	!(x==y) 为 true

条件运算符

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

语法

```
variablename=(condition)?value1:value2
```

例子

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

如果变量 visitor 中的值是 "PRES"，则向变量 greeting 赋值 "Dear President "，否则赋值 "Dear"。

JavaScript If...Else 语句

条件语句用于基于不同的条件来执行不同的动作。

条件语句

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中，我们可使用以下条件语句：

- *if* 语句 - 只有当指定条件为 true 时，使用该语句来执行代码
- *if...else* 语句 - 当条件为 true 时执行代码，当条件为 false 时执行其他代码
- *if...else if....else* 语句 - 使用该语句来选择多个代码块之一来执行
- *switch* 语句 - 使用该语句来选择多个代码块之一来执行

If 语句

只有当指定条件为 true 时，该语句才会执行代码。

语法

```
if (条件)
  {
  只有当条件为 true 时执行的代码
  }
```

注意：请使用小写的 if。使用大写字母（IF）会生成 JavaScript 错误！

实例

当时间小于 20:00 时，生成一个“Good day”问候：

```
if (time<20)
  {
  x="Good day";
  }
```

x 的结果是：

```
 <script>var d=new Date();
var time=d.getHours();
if (time<20)
  {
  document.write("Good day");
  }</script> Good day
```

请注意，在这个语法中，没有 ..else..。您已经告诉浏览器只有在指定条件为 true 时才执行代码。

If...else 语句

请使用 if....else 语句在条件为 true 时执行代码，在条件为 false 时执行其他代码。

语法

```
if (条件)
  {
  当条件为 true 时执行的代码
  }
else
  {
  当条件不为 true 时执行的代码
  }
```

实例

当时间小于 20:00 时，将得到问候 "Good day"，否则将得到问候 "Good evening"。

```
if (time<20)
  {
  x="Good day";
  }
else
  {
  x="Good evening";
  }
```

x 的结果是：

```
 <script>var d=new Date();
var time=d.getHours();
if (time<20)
  {
  document.write("Good day");
  }
else
  {
  document.write("Good evening");
  }</script> Good day
```

If...else if...else 语句

使用 if....else if...else 语句来选择多个代码块之一来执行。

语法

```
if (条件 1)
  {
  当条件 1 为 true 时执行的代码
  }
else if (条件 2)
  {
  当条件 2 为 true 时执行的代码
  }
else
  {
  当条件 1 和 条件 2 都不为 true 时执行的代码
  }
```

实例

如果时间小于 10:00，则将发送问候 "Good morning"，否则如果时间小于 20:00，则发送问候 "Good day"，否则发送问候 "Good evening"：

```
if (time<10)
  {
  x="Good morning";
  }
else if (time<20)
  {
  x="Good day";
  }
else
  {
  x="Good evening";
  }
```

x 的结果是：

```
 <script>var d=new Date();
var time=d.getHours();
if (time<10)
  {
  document.write("Good morning");
  }
else if (time<20)
  {
  document.write("Good day");
  }
else
  {
  document.write("Good evening");
  }</script> Good day
```

更多实例

随机的链接

本例将输出 W3School 或微软公司的链接。通过使用随机数，每个链接被输出的机会为 50%。

课外书

如需更多有关 *JavaScript if* 语句的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript if 语句

if 语句是 ECMAScript 中最常用的语句之一。本节为您详细讲解了如何使用 if 语句。

JavaScript Switch 语句

switch 语句用于基于不同的条件来执行不同的动作。

JavaScript Switch 语句

请使用 switch 语句来选择要执行的多个代码块之一。

语法

```
switch(n)
{
case 1:
  执行代码块 1
  break;
case 2:
  执行代码块 2
  break;
default:
  n 与 case 1 和 case 2 不同时执行的代码
}
```

工作原理：首先设置表达式 n（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 *break* 来阻止代码自动地向下一个 case 运行。

实例

显示今日的周名称。请注意 Sunday=0, Monday=1, Tuesday=2, 等等：

```
var day=new Date().getDay();
switch (day)
{
case 0:
  x="Today it's Sunday";
  break;
case 1:
  x="Today it's Monday";
  break;
case 2:
  x="Today it's Tuesday";
  break;
case 3:
  x="Today it's Wednesday";
  break;
case 4:
  x="Today it's Thursday";
  break;
case 5:
  x="Today it's Friday";
  break;
case 6:
  x="Today it's Saturday";
  break;
}
```

x 的结果：

```
 <script>var d=new Date().getDay();
switch (d)
{
case 0:
  x="Today it's Sunday";
  break;
case 1:
  x="Today it's Monday";
  break;
case 2:
  x="Today it's Tuesday";
  break;
case 3:
  x="Today it's Wednesday";
  break;
case 4:
  x="Today it's Thursday";
  break;
case 5:
  x="Today it's Friday";
  break;
case 6:
  x="Today it's Saturday";
  break;
}
document.write(x);</script> Today it's Tuesday
```

default 关键词

请使用 default 关键词来规定匹配不存在时做的事情：

实例

如果今天不是周六或周日，则会输出默认的消息：

```
var day=new Date().getDay();
switch (day)
{
case 6:
  x="Today it's Saturday";
  break;
case 0:
  x="Today it's Sunday";
  break;
`default:
  x="Looking forward to the Weekend";`
}
```

x 的结果：

```
 <script>var d=new Date().getDay();
switch (d)
{
case 6:
  x="Today it's Saturday";
  break;
case 0:
  x="Today it's Sunday";
  break;
default:
  x="Looking forward to the Weekend";
}
document.write(x);</script> Looking forward to the Weekend
```

课外书

如需更多有关 *JavaScript Switch* 语句的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript switch 语句

switch 语句是 if 语句的兄弟语句。本节介绍了 switch 语句的用法，以及与 Java 中的 switch 语句的不同。

JavaScript For 循环

循环可以将代码块执行指定的次数。

JavaScript 循环

如果您希望一遍又一遍地运行相同的代码，并且每次的值都不同，那么使用循环是很方便的。

我们可以这样输出数组的值：

```
document.write(cars[0] + "<br>");
document.write(cars[1] + "<br>");
document.write(cars[2] + "<br>");
document.write(cars[3] + "<br>");
document.write(cars[4] + "<br>");
document.write(cars[5] + "<br>");
```

不过通常我们这样写：

```
for (var i=0;i<cars.length;i++)
{
document.write(cars[i] + "<br>");
}
```

不同类型的循环

JavaScript 支持不同类型的循环：

- *for* - 循环代码块一定的次数
- *for/in* - 循环遍历对象的属性
- *while* - 当指定的条件为 true 时循环指定的代码块
- *do/while* - 同样当指定的条件为 true 时循环指定的代码块

For 循环

for 循环是您在希望创建循环时常会用到的工具。

下面是 for 循环的语法：

```
for (语句 1; 语句 2; 语句 3)
  {
  被执行的代码块
  }
```

语句 *1* 在循环（代码块）开始前执行

语句 *2* 定义运行循环（代码块）的条件

语句 *3* 在循环（代码块）已被执行之后执行

实例

```
for (var i=0; i<5; i++)
  {
  x=x + "The number is " + i + "<br>";
  }
```

从上面的例子中，您可以看到：

Statement 1 在循环开始之前设置变量 (var i=0)。

Statement 2 定义循环运行的条件（i 必须小于 5）。

Statement 3 在每次代码块已被执行后增加一个值 (i++)。

语句 1

通常我们会使用语句 1 初始化循环中所用的变量 (var i=0)。

语句 1 是可选的，也就是说不使用语句 1 也可以。

您可以在语句 1 中初始化任意（或者多个）值：

实例:

```
for (`var i=0,len=cars.length;` i<len; i++)
{
document.write(cars[i] + "<br>");
}
```

同时您还可以省略语句 1（比如在循环开始前已经设置了值时）：

实例:

```
var i=2,len=cars.length;
for (; i<len; i++)
{
document.write(cars[i] + "<br>");
}
```

语句 2

通常语句 2 用于评估初始变量的条件。

语句 2 同样是可选的。

如果语句 2 返回 true，则循环再次开始，如果返回 false，则循环将结束。

提示：如果您省略了语句 2，那么必须在循环内提供 *break*。否则循环就无法停下来。这样有可能令浏览器崩溃。请在本教程稍后的章节阅读有关 break 的内容。

语句 3

通常语句 3 会增加初始变量的值。

语句 3 也是可选的。

语句 3 有多种用法。增量可以是负数 (i--)，或者更大 (i=i+15)。

语句 3 也可以省略（比如当循环内部有相应的代码时）：

实例:

```
var i=0,len=cars.length;
for (; i<len; )
{
document.write(cars[i] + "<br>");
i++;
}
```

For/In 循环

JavaScript for/in 语句循环遍历对象的属性：

实例

```
var person={fname:"John",lname:"Doe",age:25};

for (x `in` person)
  {
  txt=txt + person[x];
  }
```

您将在有关 JavaScript 对象的章节学到更多有关 for / in 循环的知识。

While 循环

我们将在下一章为您讲解 while 循环和 do/while 循环。

课外书

如需更多有关 *JavaScript for* 语句的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 迭代语句

迭代语句又叫循环语句。本节为您介绍 ECMAScript 提供的四种迭代语句。

JavaScript While 循环

只要指定条件为 **true**，循环就可以一直执行代码。

while 循环

While 循环会在指定条件为真时循环执行代码块。

语法

```
while (条件)
  {
  需要执行的代码
  }
```

实例

本例中的循环将继续运行，只要变量 i 小于 5：

```
while (i<5)
  {
  x=x + "The number is " + i + "<br>";
  i++;
  }
```

提示：如果您忘记增加条件中所用变量的值，该循环永远不会结束。该可能导致浏览器崩溃。

do/while 循环

do/while 循环是 while 循环的变体。该循环会执行一次代码块，在检查条件是否为真之前，然后如果条件为真的话，就会重复这个循环。

语法

```
do
  {
  需要执行的代码
  }
while (条件);
```

实例

下面的例子使用 do/while 循环。该循环至少会执行一次，即使条件是 false，隐藏代码块会在条件被测试前执行：

```
do
  {
  x=x + "The number is " + i + "<br>";
  i++;
  }
while (i<5);
```

别忘记增加条件中所用变量的值，否则循环永远不会结束！

比较 for 和 while

如果您已经阅读了前面那一章关于 for 循环的内容，您会发现 while 循环与 for 循环很像。

for 语句实例

本例中的循环使用 for 循环来显示 cars 数组中的所有值：

```
cars=["BMW","Volvo","Saab","Ford"];
var i=0;
for (;cars[i];)
{
document.write(cars[i] + "<br>");
i++;
}
```

while 语句实例

本例中的循环使用使用 while 循环来显示 cars 数组中的所有值：

```
cars=["BMW","Volvo","Saab","Ford"];
var i=0;
while (cars[i])
{
document.write(cars[i] + "<br>");
i++;
}
```

课外书

如需更多有关 *JavaScript while* 语句的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 迭代语句

迭代语句又叫循环语句。本节为您介绍 ECMAScript 提供的四种迭代语句。

JavaScript Break 和 Continue 语句

break 语句用于跳出循环。

continue 用于跳过循环中的一个迭代。

Break 语句

我们已经在本教程稍早的章节中见到过 break 语句。它用于跳出 switch() 语句。

break 语句可用于跳出循环。

break 语句跳出循环后，会继续执行该循环之后的代码（如果有的话）：

实例

```
for (i=0;i<10;i++)
  {
  if (i==3)
    {
    `break;`
    }
  x=x + "The number is " + i + "<br>";
  }
```

由于这个 if 语句只有一行代码，所以可以省略花括号：

```
for (i=0;i<10;i++)
  {
  `if (i==3) break;`
  x=x + "The number is " + i + "<br>";
  }
```

Continue 语句

continue 语句中断循环中的迭代，如果出现了指定的条件，然后继续循环中的下一个迭代。

该例子跳过了值 3：

实例

```
for (i=0;i<=10;i++)
 {
 if (i==3) continue;
  x=x + "The number is " + i + "<br>";
  }
```

JavaScript 标签

正如您在 switch 语句那一章中看到的，可以对 JavaScript 语句进行标记。

如需标记 JavaScript 语句，请在语句之前加上冒号：

```
label:
语句
```

break 和 continue 语句仅仅是能够跳出代码块的语句。

语法

```
break labelname;

continue labelname;
```

continue 语句（带有或不带标签引用）只能用在循环中。

break 语句（不带标签引用），只能用在循环或 switch 中。

通过标签引用，break 语句可用于跳出任何 JavaScript 代码块：

实例

```
cars=["BMW","Volvo","Saab","Ford"];
list:
{
document.write(cars[0] + "<br>");
document.write(cars[1] + "<br>");
document.write(cars[2] + "<br>");
break list;
document.write(cars[3] + "<br>");
document.write(cars[4] + "<br>");
document.write(cars[5] + "<br>");
}
```

课外书

如需更多有关 *JavaScript Break* 和 *Continue* 语句的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript break 和 continue 语句

本节讲解了break 语句和 continue 语句的不同之处，以及如何与有标签语句一起使用。

JavaScript 错误 - Throw、Try 和 Catch

try 语句测试代码块的错误。

catch 语句处理错误。

throw 语句创建自定义错误。

错误一定会发生

当 JavaScript 引擎执行 JavaScript 代码时，会发生各种错误：

可能是语法错误，通常是程序员造成的编码错误或错别字。

可能是拼写错误或语言中缺少的功能（可能由于浏览器差异）。

可能是由于来自服务器或用户的错误输出而导致的错误。

当然，也可能是由于许多其他不可预知的因素。

JavaScript 抛出错误

当错误发生时，当事情出问题时，JavaScript 引擎通常会停止，并生成一个错误消息。

描述这种情况的技术术语是：JavaScript 将抛出一个错误。

JavaScript 测试和捕捉

try 语句允许我们定义在执行时进行错误测试的代码块。

catch 语句允许我们定义当 try 代码块发生错误时，所执行的代码块。

JavaScript 语句 *try* 和 *catch* 是成对出现的。

语法

```
try
  {
  //在这里运行代码
  }
catch(err)
  {
  //在这里处理错误
  }
```

实例

在下面的例子中，我们故意在 try 块的代码中写了一个错字。

catch 块会捕捉到 try 块中的错误，并执行代码来处理它。

```
<!DOCTYPE html>
<html>
<head>
<script>
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.message + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()">
</body>

</html>
```

Throw 语句

throw 语句允许我们创建自定义错误。

正确的技术术语是：创建或抛出异常（exception）。

如果把 throw 与 try 和 catch 一起使用，那么您能够控制程序流，并生成自定义的错误消息。

语法

```
throw _exception_
```

异常可以是 JavaScript 字符串、数字、逻辑值或对象。

实例

本例检测输入变量的值。如果值是错误的，会抛出一个异常（错误）。catch 会捕捉到这个错误，并显示一段自定义的错误消息：

```
<script>
function myFunction()
{
try
  {
  var x=document.getElementById("demo").value;
  if(x=="")    `throw "empty"`;
  if(isNaN(x)) `throw "not a number"`;
  if(x>10)     `throw "too high"`;
  if(x<5)      `throw "too low"`;
  }
catch(err)
  {
  var y=document.getElementById("mess");
  y.innerHTML="Error: " + err + ".";
  }
}
</script>

<h1>My First JavaScript</h1>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="mess"></p>
```

请注意，如果 getElementById 函数出错，上面的例子也会抛出一个错误。

课外阅读

JavaScript 高级教程： JavaScript 历史 、 JavaScript 实现

JavaScript 表单验证

JavaScript 可用来在数据被送往服务器前对 **HTML** 表单中的这些输入数据进行验证。

JavaScript 表单验证

JavaScript 可用来在数据被送往服务器前对 HTML 表单中的这些输入数据进行验证。

被 JavaScript 验证的这些典型的表单数据有：

- 用户是否已填写表单中的必填项目？
- 用户输入的邮件地址是否合法？
- 用户是否已输入合法的日期？
- 用户是否在数据域 (numeric field) 中输入了文本？

必填（或必选）项目

下面的函数用来检查用户是否已填写表单中的必填（或必选）项目。假如必填或必选项为空，那么警告框会弹出，并且函数的返回值为 false，否则函数的返回值则为 true（意味着数据没有问题）：

```
function validate_required(field,alerttxt)
{
with (field)
{
if (value==null||value=="")
  {alert(alerttxt);return false}
else {return true}
}
}
```

下面是连同 HTML 表单的代码：

```
<html>
<head>
<script type="text/javascript">
 `function validate_required(field,alerttxt)
{
with (field)
  {
  if (value==null||value=="")
    {alert(alerttxt);return false}
  else {return true}
  }
}

function validate_form(thisform)
{
with (thisform)
  {
  if (validate_required(email,"Email must be filled out!")==false)
    {email.focus();return false}
  }
}`
</script>
</head>

<body>
<form action="submitpage.htm" onsubmit="return validate_form(this)'
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>

</html>
```

E-mail 验证

下面的函数检查输入的数据是否符合电子邮件地址的基本语法。

意思就是说，输入的数据必须包含 @ 符号和点号(.)。同时，@ 不可以是邮件地址的首字符，并且 @ 之后需有至少一个点号：

```
function validate_email(field,alerttxt)
{
with (field)
{
apos=value.indexOf("@")
dotpos=value.lastIndexOf(".")
if (apos<1||dotpos-apos<2)
  {alert(alerttxt);return false}
else {return true}
}
}
```

下面是连同 HTML 表单的完整代码：

```
<html>
<head>
<script type="text/javascript">
`function validate_email(field,alerttxt)
{
with (field)
{
apos=value.indexOf("@")
dotpos=value.lastIndexOf(".")
if (apos<1||dotpos-apos<2)
  {alert(alerttxt);return false}
else {return true}
}
}

function validate_form(thisform)
{
with (thisform)
{
if (validate_email(email,"Not a valid e-mail address!")==false)
  {email.focus();return false}
}
}`
</script>
</head>

<body>
<form action="submitpage.htm"onsubmit="return validate_form(this);'
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>

</html>
```

JavaScript 保留关键字

在 JavaScript 中，一些标识符是保留关键字，不能用作变量名或函数名。

JavaScript 标准

所有的现代浏览器完全支持 ECMAScript 3（ES3，JavaScript 的第三版，从 1999 年开始）。

ECMAScript 4（ES4）未通过。

ECMAScript 5（ES5，2009 年发布），是 JavaScript 最新的官方版本。

随着时间的推移，我们开始看到，所有的现代浏览器已经完全支持 ES5。

JavaScript 保留关键字

Javascript 的保留关键字不可以用作变量、标签或者函数名。有些保留关键字是作为 Javascript 以后扩展使用。

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

- 标记的关键字是 ECMAScript5 中新添加的。

JavaScript 对象、属性和方法

您也应该避免使用 JavaScript 内置的对象、属性和方法的名称作为 Javascript 的变量或函数名：

Array	Date	eval	function	hasOwnProperty
Infinity	isFinite	isNaN	isPrototypeOf	length
Math	NaN	name	Number	Object
prototype	String	toString	undefined	valueOf

Java 保留关键字

JavaScript 经常与 Java 一起使用。您应该避免使用一些 Java 对象和属性作为 JavaScript 标识符：

getClass	java	JavaArray
javaClass	JavaObject	JavaPackage

Windows 保留关键字

JavaScript 可以在 HTML 外部使用。它可在许多其他应用程序中作为编程语言使用。

在 HTML 中，您必须（为了可移植性，您也应该这么做）避免使用 HTML 和 Windows 对象和属性的名称作为 Javascript 的变量及函数名：

alert	all	anchor	anchors
assign	blur	button	checkbox
clearTimeout	clientInformation	close	closed
constructor	crypto	decodeURI	decodeURICompo
document	element	elements	embed
encodeURI	encodeURIComponent	escape	event
focus	form	forms	frame
innerWidth	layer	layers	link
mimeTypes	navigate	navigator	frames
hidden	history	image	images
open	opener	option	outerHeight
packages	pageXOffset	pageYOffset	parent
parseInt	password	pkcs11	plugin
propertyIsEnum	radio	reset	screenX
scroll	secure	select	self
setTimeout	status	submit	taint
textarea	top	unescape	untaint

HTML 事件句柄

除此之外，您还应该避免使用 HTML 事件句柄的名称作为 Javascript 的变量及函数名。

实例：

onblur	onclick	onerror	onfocus
onkeydown	onkeypress	onkeyup	onmouseover
onload	onmouseup	onmousedown	onsubmit

非标准 JavaScript

除了保留关键字，在 JavaScript 实现中也有一些非标准的关键字。

一个实例是 **const** 关键字，用于定义变量。 一些 JavaScript 引擎把 const 当作 var 的同义词。另一些引擎则把 const 当作只读变量的定义。

Const 是 JavaScript 的扩展。JavaScript 引擎支持它用在 Firefox 和 Chrome 中。但是它并不是 JavaScript 标准 ES3 或 ES5 的组成部分。建议：不要使用它。

JavaScript JSON

JSON 是用于存储和传输数据的格式。

JSON 通常用于服务端向网页传递数据 。

什么是 JSON?

- JSON 英文全称 **J**ava**S**cript **O**bject **N**otation
- JSON 是一种轻量级的数据交换格式。
- JSON是独立的语言 *
- JSON 易于理解。

> JSON 使用 JavaScript 语法，但是 JSON 格式仅仅是一个文本。
> 文本可以被任何编程语言读取及作为数据格式传递。

JSON 实例

以下 JSON 语法定义了 employees 对象: 3 条员工记录（对象）的数组:

JSON Example

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

JSON 格式化后为 JavaScript 对象

JSON 格式在语法上与创建 JavaScript 对象代码是相同的。

由于它们很相似，所以 JavaScript 程序可以很容易的将 JSON 数据转换为 JavaScript 对象。

JSON 语法规则

- 数据为 键/值 对。
- 数据由逗号分隔。

- 大括号保存对象
- 方括号保存数组

JSON 数据 - 一个名称对应一个值

JSON 数据格式为 键/值 对，就像 JavaScript 对象属性。

键/值对包括字段名称（在双引号中），后面一个冒号，然后是值：

```
"firstName":"John"
```

JSON 对象

JSON 对象保存在大括号内。

就像在 JavaScript 中, 对象可以保存多个 键/值 对：

```
{"firstName":"John", "lastName":"Doe"}
```

JSON 数组

JSON 数组保存在中括号内。

就像在 JavaScript 中, 数组可以包含对象：

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
```

在以上实例中，对象 "employees" 是一个数组。包含了三个对象。

每个为个对象为员工的记录（姓和名）。

JSON 字符串转换为 JavaScript 对象

通常我们从服务器中读取 JSON 数据，并在网页中显示数据。

简单起见，我们网页中直接设置 JSON 字符串 (你还可以阅读我们的 JSON 教程):

首先，创建 JavaScript 字符串，字符传为 JSON 格式的数据：

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

然后，使用 JavaScript 内置函数 JSON.parse() 将字符串转换为 JavaScript 对象:

```
var obj = JSON.parse(text);
```

最后，在你的页面中使用新的 JavaScript 对象：

实例

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

更多 JSON 信息，你可以阅读我们的 JSON 教程。

javascript:void(0) 含义

我们经常会使用到 javascript:void(0) 这样的代码，那么在 JavaScript 中 javascript:void(0) 代表的是什么意思呢？

javascript:void(0) 中最关键的是 void 关键字， void 是 JavaScript 中非常重要的关键字，该操作符指定要计算一个表达式但是不返回值。

语法格式如下：

```
<head>
<script type="text/javascript">
<!--
void func()
javascript:void func()

或者

void(func())
javascript:void(func())
//-->
</script>
</head>
```

下面的代码创建了一个超级链接，当用户点击以后不会发生任何事。

实例

```
<a href="javascript:void(0)">单击此处什么也不会发生</a>
```

当用户链接时，void(0) 计算为 0，但 Javascript 上没有任何效果。

以下实例中，在用户点击链接后显示警告信息：

实例

```
<head>
<script type="text/javascript">
<!--
//-->
</script>
</head>
<body>
<a href="javascript:void(alert('Warning!!!'))">点我!</a>
</body>
```

以下实例中参数 a 将返回 undefined :

实例

```
<head>
<script type="text/javascript">
<!--
function getValue(){
  var a,b,c;
  a = void ( b = 5, c = 7 );
  document.write('a = ' + a + ' b = ' + b +' c = ' + c );
}
//-->
</script>
</head>
```

href="#"与href="javascript:void(0)"的区别

包含了一个位置信息，默认的锚是**#top** 也就是网页的上端。

而javascript:void(0), 仅仅表示一个死链接。

在页面很长的时候会使用 **#** 来定位页面的具体位置，格式为：**# + id**。

如果你要定义一个死链接请使用 javascript:void(0) 。

实例

```
<a href="javascript:void(0);">点我没有反应的!</a>
<a href="#pos">点我定位到指定位置!</a>
<br><br><br> <p id="pos">尾部定位点</p>
```

JavaScript 高级

JavaScript 对象

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...

此外，**JavaScript 允许自定义对象。**

JavaScript 对象

JavaScript 提供多个内建对象，比如 String、Date、Array 等等。

对象只是带有属性和方法的特殊数据类型。

访问对象的属性

属性是与对象相关的值。

访问对象属性的语法是：

```
_objectName_._propertyName_
```

这个例子使用了 String 对象的 length 属性来获得字符串的长度：

```
var message="Hello World!";
var x=message.`length`;
```

在以上代码执行后，x 的值将是：

```
12
```

访问对象的方法

方法是能够在对象上执行的动作。

您可以通过以下语法来调用方法：

```
_objectName_._methodName_()
```

这个例子使用了 String 对象的 toUpperCase() 方法来将文本转换为大写：

```
var message="Hello world!";
var x=message.`toUpperCase()`;
```

在以上代码执行后，x 的值将是：

```
HELLO WORLD!
```

创建 JavaScript 对象

通过 JavaScript，您能够定义并创建自己的对象。

创建新对象有两种不同的方法：

1. 定义并创建对象的实例
2. 使用函数来定义对象，然后创建新的对象实例

创建直接的实例

这个例子创建了对象的一个新实例，并向其添加了四个属性：

实例

```
person=new Object();
person.firstname="Bill";
person.lastname="Gates";
person.age=56;
person.eyecolor="blue";
```

替代语法（使用对象 literals）：

实例

```
person={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};
```

使用对象构造器

本例使用函数来构造对象：

实例

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;
}
```

创建 JavaScript 对象实例

一旦您有了对象构造器，就可以创建新的对象实例，就像这样：

```
var myFather=new person("Bill","Gates",56,"blue");
var myMother=new person("Steve","Jobs",48,"green");
```

把属性添加到 JavaScript 对象

您可以通过为对象赋值，向已有对象添加新属性：

假设 personObj 已存在 - 您可以为其添加这些新属性：firstname、lastname、age 以及 eyecolor：

```
person.firstname="Bill";
person.lastname="Gates";
person.age=56;
person.eyecolor="blue";

x=person.firstname;
```

在以上代码执行后，x 的值将是：

```
Bill
```

把方法添加到 JavaScript 对象

方法只不过是附加在对象上的函数。

在构造器函数内部定义对象的方法：

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;

this.changeName=changeName;
function changeName(name)
{
this.lastname=name;
}
}
```

changeName() 函数 name 的值赋给 person 的 lastname 属性。

现在您可以试一下：

```
myMother.changeName("Ballmer");
```

JavaScript 类

JavaScript 是面向对象的语言，但 JavaScript 不使用类。

在 JavaScript 中，不会创建类，也不会通过类来创建对象（就像在其他面向对象的语言中那样）。

JavaScript 基于 prototype，而不是基于类的。

JavaScript for...in 循环

JavaScript for...in 语句循环遍历对象的属性。

语法

```
for (对象中的变量)
  {
  要执行的代码
  }
```

注释：for...in 循环中的代码块将针对每个属性执行一次。

实例

循环遍历对象的属性：

```
var person={fname:"Bill",lname:"Gates",age:56};

for (x in person)
  {
  txt=txt + person[x];
  }
```

课外书

如需更多有关 *JavaScript* 对象的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 面向对象技术

本节简要介绍了面向对象技术的术语、面向对象语言的要求以及对象的构成。

ECMAScript 对象应用

本节讲解了如何声明和实例化对象，如何引用和废除对象，以及绑定的概念。

ECMAScript 对象类型

本节介绍了 ECMAScript 的三种类型：本地对象、内置对象和宿主对象，并提供了指向相关参考手册的链接。

ECMAScript 对象作用域

本节讲解了 ECMAScript 作用域以及 this 关键字。

ECMAScript 定义类或对象

本节详细讲解了创建 ECMAScript 对象或类的各种方式。

ECMAScript 修改对象

本节讲解了如何通过创建新方法或重定义已有方法来修改对象。

JavaScript Number 对象

JavaScript 只有一种数字类型。

可以使用也可以不使用小数点来书写数字。

JavaScript 数字

JavaScript 数字可以使用也可以不使用小数点来书写：

实例

```
var pi=3.14;    // 使用小数点
var x=34;       // 不使用小数点
```

极大或极小的数字可通过科学（指数）计数法来写：

实例

```
var y=123e5;    // 12300000
var z=123e-5;   // 0.00123
```

所有 JavaScript 数字均为 64 位

JavaScript 不是类型语言。与许多其他编程语言不同，JavaScript 不定义不同类型的数字，比如整数、短、长、浮点等等。

JavaScript 中的所有数字都存储为根为 10 的 64 位（8 比特），浮点数。

精度

整数（不使用小数点或指数计数法）最多为 15 位。

小数的最大位数是 17，但是浮点运算并不总是 100% 准确：

实例

```
var x=0.2+0.1;
```

八进制和十六进制

如果前缀为 0，则 JavaScript 会把数值常量解释为八进制数，如果前缀为 0 和 "x"，则解释为十六进制数。

实例

```
var y=0377;
var z=0xFF;
```

提示：绝不要在数字前面写零，除非您需要进行八进制转换。

数字属性和方法

属性：

- MAX VALUE
- MIN VALUE
- NEGATIVE INFINITIVE
- POSITIVE INFINITIVE
- NaN
- prototype
- constructor

方法：

- toExponential()
- toFixed()
- toPrecision()
- toString()
- valueOf()

完整的 Number 对象参考手册

如需可用于 Number 对象的所有属性和方法的完整参考，请访问我们的 Number 对象参考手册。

该参考手册包含每个属性和方法的描述和实例。

JavaScript 字符串(String)对象

String 对象用于处理已有的字符块。

JavaScript String（字符串）对象 实例

计算字符串的长度

如何使用长度属性来计算字符串的长度。

```
<html>
<body>

<script type="text/javascript">

var txt="Hello World!"
document.write(txt.length)

</script>

</body>
</html>
```

为字符串添加样式

如何为字符串添加样式。

```
<html>
<body>

<script type="text/javascript">

var txt="Hello World!"

document.write("<p>Big: " + txt.big() + "</p>")
document.write("<p>Small: " + txt.small() + "</p>")

document.write("<p>Bold: " + txt.bold() + "</p>")
document.write("<p>Italic: " + txt.italics() + "</p>")

document.write("<p>Blink: " + txt.blink() + " (does not work in IE)
document.write("<p>Fixed: " + txt.fixed() + "</p>")
document.write("<p>Strike: " + txt.strike() + "</p>")

document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>")
document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>")

document.write("<p>Lowercase: " + txt.toLowerCase() + "</p>")
document.write("<p>Uppercase: " + txt.toUpperCase() + "</p>")

document.write("<p>Subscript: " + txt.sub() + "</p>")
document.write("<p>Superscript: " + txt.sup() + "</p>")

document.write("<p>Link: " + txt.link("http://www.w3school.com.cn")
</script>

</body>
</html>
```

indexOf() 方法

如何使用 indexOf() 来定位字符串中某一个指定的字符首次出现的位置。

```
<html>
<body>

<script type="text/javascript">

var str="Hello world!"
document.write(str.indexOf("Hello") + "<br />")
document.write(str.indexOf("World") + "<br />")
document.write(str.indexOf("world"))

</script>

</body>
</html>
```

match() 方法

如何使用 match() 来查找字符串中特定的字符，并且如果找到的话，则返回这个字符。

```
<html>
<body>

<script type="text/javascript">

var str="Hello world!"
document.write(str.match("world") + "<br />")
document.write(str.match("World") + "<br />")
document.write(str.match("worlld") + "<br />")
document.write(str.match("world!"))

</script>

</body>
</html>
```

如何替换字符串中的字符 - replace()

如何使用 replace() 方法在字符串中用某些字符替换另一些字符。

```
<html>
<body>

<script type="text/javascript">

var str="Visit Microsoft!"
document.write(str.replace(/Microsoft/,"W3School"))

</script>
</body>
</html>
```

完整的 String 对象参考手册

请查看我们的 JavaScript String 对象参考手册，其中提供了可以与字符串对象一同使用的所有的属性和方法。

这个手册包含的关于每个属性和方法的用法的详细描述和实例。

字符串对象

字符串对象用于处理已有的字符块。

例子：

下面的例子使用字符串对象的长度属性来计算字符串的长度。

```
var txt="Hello world!"
document.write(txt.`length`)
```

上面的代码输出为：

```
12
```

下面的例子使用字符串对象的toUpperCase()方法将字符串转换为大写：

```
var txt="Hello world!"
document.write(txt.`toUpperCase()`)
```

上面的代码输出为：

```
HELLO WORLD!
```

相关页面

JavaScript 高级教程：ECMAScript 类型转换

JavaScript 高级教程：ECMAScript 引用类型

JavaScript 参考手册：String 对象

JavaScript Date（日期）对象

日期对象用于处理日期和时间。

JavaScript Date（日期）对象 实例

返回当日的日期和时间

如何使用 Date() 方法获得当日的日期。

```
<html>
<body>

<script type="text/javascript">

document.write(Date())

</script>

</body>
</html>
```

getTime()

getTime() 返回从 1970 年 1 月 1 日至今的毫秒数。

```
<html>
<body>

<script type="text/javascript">
var d=new Date();
document.write("从 1970/01/01 至今已过去 " + d.getTime() + " 毫秒");
</script>

</body>
</html>
```

setFullYear()

如何使用 setFullYear() 设置具体的日期。

```
<html>
<body>

<script type="text/javascript">

var d = new Date()
d.setFullYear(1992,10,3)
document.write(d)

</script>

</body>
</html>
```

toUTCString()

如何使用 toUTCString() 将当日的日期（根据 UTC）转换为字符串。

```
<html>
<body>

<script type="text/javascript">

var d = new Date()
document.write (d.toUTCString())

</script>

</body>
</html>
```

getDay()

如何使用 getDay() 和数组来显示星期，而不仅仅是数字。

```
<html>
<body>

<script type="text/javascript">

var d=new Date()
var weekday=new Array(7)
weekday[0]="星期日"
weekday[1]="星期一"
weekday[2]="星期二"
weekday[3]="星期三"
weekday[4]="星期四"
weekday[5]="星期五"
weekday[6]="星期六"

document.write("今天是" + weekday[d.getDay()])

</script>

</body>
</html>
```

显示一个钟表

如何在网页上显示一个钟表。

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date()
var h=today.getHours()
var m=today.getMinutes()
var s=today.getSeconds()
// add a zero in front of numbers<10
m=checkTime(m)
s=checkTime(s)
document.getElementById('txt').innerHTML=h+":"+m+":"+s
t=setTimeout('startTime()',500)
}

function checkTime(i)
{
if (i<10)
  {i="0" + i}
  return i
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

完整的 Date 对象参考手册

我们提供 JavaScript Date 对象参考手册，其中包括所有可用于日期对象的属性和方法。

该手册包含了对每个属性和方法的详细描述以及相关实例。

定义日期

Date 对象用于处理日期和时间。

可以通过 new 关键词来定义 Date 对象。以下代码定义了名为 myDate 的 Date 对象：

```
var myDate=`new` Date()
```

注释：Date 对象自动使用当前的日期和时间作为其初始值。

操作日期

通过使用针对日期对象的方法，我们可以很容易地对日期进行操作。

在下面的例子中，我们为日期对象设置了一个特定的日期 (2008 年 8 月 9 日)：

```
var myDate=new Date()
myDate.`setFullYear`(2008,7,9)
```

注意：表示月份的参数介于 0 到 11 之间。也就是说，如果希望把月设置为 8 月，则参数应该是 7。

在下面的例子中，我们将日期对象设置为 5 天后的日期：

```
var myDate=new Date()
myDate.`setDate`(myDate.getDate()+5)
```

注意：如果增加天数会改变月份或者年份，那么日期对象会自动完成这种转换。

比较日期

日期对象也可用于比较两个日期。

下面的代码将当前日期与 2008 年 8 月 9 日做了比较：

```
var myDate=new Date();
myDate.setFullYear(2008,8,9);

var today = new Date();

if (myDate>today)
{
alert("Today is before 9th August 2008");
}
else
{
alert("Today is after 9th August 2008");
}
```

JavaScript Array（数组）对象

数组对象的作用是：使用单独的变量名来存储一系列的值。

实例

创建数组

创建数组，为其赋值，然后输出这些值。

```
<html>
<body>

<script type="text/javascript">
var mycars = new Array()
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"

for (i=0;i<mycars.length;i++)
{
document.write(mycars[i] + "<br />")
}
</script>

</body>
</html>
```

For...In 声明

使用 for...in 声明来循环输出数组中的元素。

```
<html>
<body>
<script type="text/javascript">
var x
var mycars = new Array()
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"

for (x in mycars)
{
document.write(mycars[x] + "<br />")
}
</script>
</body>
</html>
```

合并两个数组 - concat()

如何使用 concat() 方法来合并两个数组。

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

var arr2 = new Array(3)
arr2[0] = "James"
arr2[1] = "Adrew"
arr2[2] = "Martin"

document.write(arr.concat(arr2))

</script>

</body>
</html>
```

用数组的元素组成字符串 - join()

如何使用 join() 方法将数组的所有元素组成一个字符串。

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(3);
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr.join());

document.write("<br />");

document.write(arr.join("."));

</script>

</body>
</html>
```

文字数组 - sort()

如何使用 sort() 方法从字面上对数组进行排序。

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
document.write(arr.sort())

</script>

</body>
</html>
```

数字数组 - sort()

如何使用 sort() 方法从数值上对数组进行排序。

```
<html>
<body>

<script type="text/javascript">

function sortNumber(a, b)
{
return a - b
}

var arr = new Array(6)
arr[0] = "10"
arr[1] = "5"
arr[2] = "40"
arr[3] = "25"
arr[4] = "1000"
arr[5] = "1"

document.write(arr + "<br />")
document.write(arr.sort(sortNumber))

</script>

</body>
</html>
```

完整的 Array 对象参考手册

我们提供 JavaScript Array 对象参考手册，其中包括所有可用于数组对象的属性和方法。

该手册包含了对每个属性和方法的详细描述以及相关实例。

定义数组

数组对象用来在单独的变量名中存储一系列的值。

我们使用关键词 new 来创建数组对象。下面的代码定义了一个名为 myArray 的数组对象：

```
var myArray=new Array()
```

有两种向数组赋值的方法（你可以添加任意多的值，就像你可以定义你需要的任意多的变量一样）。

1:

```
var mycars=`new Array()`
mycars[0]="Saab"
mycars[1]="Volvo"
mycars[2]="BMW"
```

也可以使用一个整数自变量来控制数组的容量：

```
var mycars=`new Array(3)`
mycars[0]="Saab"
mycars[1]="Volvo"
mycars[2]="BMW"
```

2:

```
var mycars=`new Array("Saab","Volvo","BMW")`
```

注意：如果你需要在数组内指定数值或者逻辑值，那么变量类型应该是数值变量或者布尔变量，而不是字符变量。

访问数组

通过指定数组名以及索引号码，你可以访问某个特定的元素。

下面是代码行：

```
document.write(mycars[0])
```

下面是输出：

```
Saab
```

修改已有数组中的值

如需修改已有数组中的值，只要向指定下标号添加一个新值即可：

```
mycars[0]="Opel";
```

现在，以上代码：

```
document.write(mycars[0]);
```

将输出：

```
Opel
```

JavaScript Boolean（逻辑）对象

Boolean（逻辑）对象用于将非逻辑值转换为逻辑值（**true** 或者 **false**）。

实例

检查逻辑值

检查逻辑对象是 true 还是 false。

```
<html>
<body>

<script type="text/javascript">
var b1=new Boolean( 0)
var b2=new Boolean(1)
var b3=new Boolean("")
var b4=new Boolean(null)
var b5=new Boolean(NaN)
var b6=new Boolean("false")

document.write("0 是逻辑的 "+ b1 +"<br />")
document.write("1 是逻辑的 "+ b2 +"<br />")
document.write("空字符串是逻辑的 "+ b3 + "<br />")
document.write("null 是逻辑的 "+ b4+ "<br />")
document.write("NaN 是逻辑的 "+ b5 +"<br />")
document.write("字符串 'false' 是逻辑的 "+ b6 +"<br />")
</script>

</body>
</html>
```

完整的 Boolean 对象参考手册

我们提供 JavaScript Boolean 对象参考手册，其中包括所有可用于逻辑对象的属性和方法。

该手册包含了对每个属性和方法的详细描述以及相关实例。

Boolean 对象

您可以将 Boolean 对象理解为一个产生逻辑值的对象包装器。

Boolean（逻辑）对象用于将非逻辑值转换为逻辑值（true 或者 false）。

创建 Boolean 对象

使用关键词 new 来定义 Boolean 对象。下面的代码定义了一个名为 myBoolean 的逻辑对象：

```
var myBoolean=new Boolean()
```

注释：如果逻辑对象无初始值或者其值为 0、-0、null、""、false、undefined 或者 NaN，那么对象的值为 false。否则，其值为 true（即使当自变量为字符串 "false" 时）！

下面的所有的代码行均会创建初始值为 false 的 Boolean 对象。

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

下面的所有的代码行均会创初始值为 true 的 Boolean 对象：

```
var myBoolean=new Boolean(1);
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Bill Gates");
```

相关页面

JavaScript 高级教程：ECMAScript 引用类型

JavaScript 参考手册：Boolean 对象

JavaScript Math（算数）对象

Math（算数）对象的作用是：执行常见的算数任务。

实例

round()

如何使用 round()。

```
<html>
<body>

<script type="text/javascript">

document.write(Math.round(0.60) + "<br />")
document.write(Math.round(0.50) + "<br />")
document.write(Math.round(0.49) + "<br />")
document.write(Math.round(-4.40) + "<br />")
document.write(Math.round(-4.60))

</script>

</body>
</html>
```

random()

如何使用 random() 来返回 0 到 1 之间的随机数。

```
<html>
<body>

<script type="text/javascript">

document.write(Math.random())

</script>

</body>
</html>
```

max()

如何使用 max() 来返回两个给定的数中的较大的数。（在 ECMASCript v3 之前，该方法只有两个参数。）

```
<html>
<body>

<script type="text/javascript">

document.write(Math.max(5,7) + "<br />")
document.write(Math.max(-3,5) + "<br />")
document.write(Math.max(-3,-5) + "<br />")
document.write(Math.max(7.25,7.30))

</script>

</body>
</html>
```

min()

如何使用 min() 来返回两个给定的数中的较小的数。（在 ECMASCript v3 之前，该方法只有两个参数。）

```
<html>
<body>

<script type="text/javascript">

document.write(Math.min(5,7) + "<br />")
document.write(Math.min(-3,5) + "<br />")
document.write(Math.min(-3,-5) + "<br />")
document.write(Math.min(7.25,7.30))

</script>

</body>
</html>
```

完整的 Math 对象参考手册

我们提供 JavaScript Math 对象的参考手册，其中包括所有可用于算术对象的属性和方法。

该手册包含了对每个属性和方法的详细描述以及相关实例。

Math 对象

Math（算数）对象的作用是：执行普通的算数任务。

Math 对象提供多种算数值类型和函数。无需在使用这个对象之前对它进行定义。

算数值

JavaScript 提供 8 种可被 Math 对象访问的算数值：

- 常数
- 圆周率
- 2 的平方根
- 1/2 的平方根
- 2 的自然对数
- 10 的自然对数
- 以 2 为底的 e 的对数
- 以 10 为底的 e 的对数

这是在 Javascript 中使用这些值的方法：（与上面的算数值一一对应）

- Math.E
- Math.PI
- Math.SQRT2
- Math.SQRT1_2
- Math.LN2
- Math.LN10
- Math.LOG2E
- Math.LOG10E

算数方法

除了可被 Math 对象访问的算数值以外，还有几个函数（方法）可以使用。

函数（方法）实例：

下面的例子使用了 Math 对象的 round 方法对一个数进行四舍五入。

```
document.write(`Math.round(4.7)`)
```

上面的代码输出为：

```
5
```

下面的例子使用了 Math 对象的 random() 方法来返回一个介于 0 和 1 之间的随机数：

```
document.write(`Math.random()`)
```

上面的代码输出为：

```
0.9370844220218102
```

下面的例子使用了 Math 对象的 floor() 方法和 random() 来返回一个介于 0 和 10 之间的随机数：

```
document.write(`Math.floor(Math.random()*11)`)
```

上面的代码输出为：

```
3
```

JavaScript RegExp 对象

RegExp 对象用于规定在文本中检索的内容。

什么是 RegExp？

RegExp 是正则表达式的缩写。

当您检索某个文本时，可以使用一种模式来描述要检索的内容。RegExp 就是这种模式。

简单的模式可以是一个单独的字符。

更复杂的模式包括了更多的字符，并可用于解析、格式检查、替换等等。

您可以规定字符串中的检索位置，以及要检索的字符类型，等等。

定义 RegExp

RegExp 对象用于存储检索模式。

通过 new 关键词来定义 RegExp 对象。以下代码定义了名为 patt1 的 RegExp 对象，其模式是 "e"：

```
var patt1=new RegExp("e");
```

当您使用该 RegExp 对象在一个字符串中检索时，将寻找的是字符 "e"。

RegExp 对象的方法

RegExp 对象有 3 个方法：test()、exec() 以及 compile()。

test()

test() 方法检索字符串中的指定值。返回值是 true 或 false。

例子：

```
var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));
```

由于该字符串中存在字母 "e"，以上代码的输出将是：

```
true
```

TIY

exec()

exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。

例子 1：

```
var patt1=new RegExp("e");

document.write(patt1.exec("The best things in life are free"));
```

由于该字符串中存在字母 "e"，以上代码的输出将是：

```
e
```

TIY

例子 2：

您可以向 RegExp 对象添加第二个参数，以设定检索。例如，如果需要找到所有某个字符的所有存在，则可以使用 "g" 参数 ("global")。

如需关于如何修改搜索模式的完整信息，请访问我们的 RegExp 对象参考手册。

在使用 "g" 参数时，exec() 的工作原理如下：

- 找到第一个 "e"，并存储其位置
- 如果再次运行 exec()，则从存储的位置开始检索，并找到下一个 "e"，并存储其位置

```
var patt1=new RegExp("e","g");
do
{
result=patt1.exec("The best things in life are free");
document.write(result);
}
while (result!=null)
```

由于这个字符串中 6 个 "e" 字母，代码的输出将是：

```
eeeeeenull
```

TIY

compile()

compile() 方法用于改变 RegExp。

compile() 既可以改变检索模式，也可以添加或删除第二个参数。

例子：

```
var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

patt1.compile("d");

document.write(patt1.test("The best things in life are free"));
```

由于字符串中存在 "e"，而没有 "d"，以上代码的输出是：

```
truefalse
```

TIY

完整的 RegExp 对象参考手册

如需可与 RegExp 对象一同使用所有属性和方法的完整信息，请访问我们的 RegExp 对象参考手册。

这个参考手册包含了对 RegExp 对象中每个属性和方法的详细描述，以及使用的例子。

JavaScript BOM

JavaScript Window - 浏览器对象模型

浏览器对象模型 (**BOM**) 使 **JavaScript** 有能力与浏览器“对话”。

浏览器对象模型 (BOM)

浏览器对象模型（Browser Object Model）尚无正式标准。

由于现代浏览器已经（几乎）实现了 JavaScript 交互性方面的相同方法和属性，因此常被认为是 BOM 的方法和属性。

Window 对象

所有浏览器都支持 *window* 对象。它表示浏览器窗口。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。

全局变量是 window 对象的属性。

全局函数是 window 对象的方法。

甚至 HTML DOM 的 document 也是 window 对象的属性之一：

```
window.document.getElementById("header");
```

与此相同：

```
document.getElementById("header");
```

Window 尺寸

有三种方法能够确定浏览器窗口的尺寸（浏览器的视口，不包括工具栏和滚动条）。

对于Internet Explorer、Chrome、Firefox、Opera 以及 Safari：

- window.innerHeight - 浏览器窗口的内部高度
- window.innerWidth - 浏览器窗口的内部宽度

对于 Internet Explorer 8、7、6、5：

- document.documentElement.clientHeight
- document.documentElement.clientWidth

或者

- document.body.clientHeight
- document.body.clientWidth

实用的 JavaScript 方案（涵盖所有浏览器）：

实例

```
var w=window.innerWidth
|| document.documentElement.clientWidth
|| document.body.clientWidth;

var h=window.innerHeight
|| document.documentElement.clientHeight
|| document.body.clientHeight;
```

该例显示浏览器窗口的高度和宽度：（不包括工具栏/滚动条）

其他 Window 方法

一些其他方法：

- window.open() - 打开新窗口
- window.close() - 关闭当前窗口
- window.moveTo() - 移动当前窗口
- window.resizeTo() - 调整当前窗口的尺寸

JavaScript Window Screen

window.screen 对象包含有关用户屏幕的信息。

Window Screen

window.screen 对象在编写时可以不使用 window 这个前缀。

一些属性：

- screen.availWidth - 可用的屏幕宽度
- screen.availHeight - 可用的屏幕高度

Window Screen 可用宽度

screen.availWidth 属性返回访问者屏幕的宽度，以像素计，减去界面特性，比如窗口任务栏。

实例

返回您的屏幕的可用宽度：

```
<script>

document.write("可用宽度：" + `screen.availWidth`);

</script>
```

以上代码输出为：

```
<script>document.write("可用宽度：" + screen.availWidth);</script>可
```

Window Screen 可用高度

screen.availHeight 属性返回访问者屏幕的高度，以像素计，减去界面特性，比如窗口任务栏。

实例

返回您的屏幕的可用高度：

```
<script>

document.write("可用高度：" + `screen.availHeight`);

</script>
```

以上代码输出为：

```
<script>document.write("可用高度：" + screen.availHeight);</script>可
```

JavaScript Window Location

window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

Window Location

window.location 对象在编写时可不使用 window 这个前缀。

一些例子：

- location.hostname 返回 web 主机的域名
- location.pathname 返回当前页面的路径和文件名
- location.port 返回 web 主机的端口 （80 或 443）
- location.protocol 返回所使用的 web 协议（http:// 或 https://）

Window Location Href

location.href 属性返回当前页面的 URL。

实例

返回（当前页面的）整个 URL：

```
<script>

document.write(`location.href`);

</script>
```

以上代码输出为：

```
<script>document.write(location.href);</script>http://www.w3school
```

Window Location Pathname

location.pathname 属性返回 URL 的路径名。

实例

返回当前 URL 的路径名：

```
<script>

document.write(`location.pathname`);

</script>
```

以上代码输出为：

```
<script>document.write(location.pathname);</script>/js/js_window_lo
```

Window Location Assign

location.assign() 方法加载新的文档。

实例

加载一个新的文档：

```
<html>
<head>
<script>
function newDoc()
  {
  `window.location.assign("http://www.w3school.com.cn")`
  }
</script>
</head>
<body>

<input type="button" value="加载新文档" onclick="newDoc()">

</body>
</html>
```

JavaScript Window History

window.history 对象包含浏览器的历史。

Window History

window.history 对象在编写时可不使用 window 这个前缀。

为了保护用户隐私，对 JavaScript 访问该对象的方法做出了限制。

一些方法：

- history.back() - 与在浏览器点击后退按钮相同
- history.forward() - 与在浏览器中点击按钮向前相同

Window History Back

history.back() 方法加载历史列表中的前一个 URL。

这与在浏览器中点击后退按钮是相同的：

实例

在页面上创建后退按钮：

```
<html>
<head>
<script>
function goBack()
  {
  `window.history.back()`
  }
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

以上代码输出为：

```
<input type="button" value="转到上一页" onclick="goBack()">
```

Window History Forward

history forward() 方法加载历史列表中的下一个 URL。

这与在浏览器中点击前进按钮是相同的：

实例

在页面上创建一个向前的按钮：

```
<html>
<head>
<script>
function goForward()
  {
  `window.history.forward()`
  }
</script>
</head>
<body>

<input type="button" value="Forward" onclick="goForward()">

</body>
</html>
```

以上代码输出为：

```
<input type="button" value="转到下一页" onclick="goForward()">
```

JavaScript Window Navigator

window.navigator 对象包含有关访问者浏览器的信息。

Window Navigator

window.navigator 对象在编写时可不使用 window 这个前缀。

实例

```
<div id="example"></div>

<script>

txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>

document.getElementById("example").innerHTML=txt;

</script>
```

警告：来自 navigator 对象的信息具有误导性，不应该被用于检测浏览器版本，这是因为：

- navigator 数据可被浏览器使用者更改
- 浏览器无法报告晚于浏览器发布的新操作系统

浏览器检测

由于 navigator 可误导浏览器检测，使用对象检测可用来嗅探不同的浏览器。

由于不同的浏览器支持不同的对象，您可以使用对象来检测浏览器。例如，由于只有 Opera 支持属性 "window.opera"，您可以据此识别出 Opera。

例子：if (window.opera) {...some action...}

JavaScript 消息框

可以在 **JavaScript** 中创建三种消息框：警告框、确认框、提示框。

实例

警告框

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("我是警告框！！")
}
</script>
</head>
<body>

<input type="button" onclick="disp_alert()" value="显示警告框" />

</body>
</html>
```

带有折行的警告框

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("再次向您问好！在这里，我们向您演示" + '\n' + "如何向警告框添加折行。"
}
</script>
</head>
<body>

<input type="button" onclick="disp_alert()" value="显示警告框" />

</body>
</html>
```

确认框

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button!");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show a confirr

</body>
</html>
```

提示框

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
  {
  var name=prompt("请输入您的名字","Bill Gates")
  if (name!=null && name!="")
    {
    document.write("你好！" + name + " 今天过得怎么样？")
    }
  }
</script>
</head>
<body>

<input type="button" onclick="disp_prompt()" value="显示提示框" />

</body>
</html>
```

警告框

警告框经常用于确保用户可以得到某些信息。

当警告框出现后，用户需要点击确定按钮才能继续进行操作。

语法：

```
alert("文本")
```

确认框

确认框用于使用户可以验证或者接受某些信息。

当确认框出现后，用户需要点击确定或者取消按钮才能继续进行操作。

如果用户点击确认，那么返回值为 true。如果用户点击取消，那么返回值为 false。

语法：

```
confirm("文本")
```

提示框

提示框经常用于提示用户在进入页面前输入某个值。

当提示框出现后，用户需要输入某个值，然后点击确认或取消按钮才能继续操纵。

如果用户点击确认，那么返回值为输入的值。如果用户点击取消，那么返回值为 null。

语法：

```
prompt("文本","默认值")
```

JavaScript 计时

通过使用 **JavaScript**，我们有能力做到在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。我们称之为计时事件。

实例

简单的计时

单击本例中的按钮后，会在 5 秒后弹出一个警告框。

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 秒！')",5000)
}
</script>
</head>

<body>
<form>
<input type="button" value="显示定时的警告框" onClick = "timedMsg()">
</form>
<p>请点击上面的按钮。警告框会在 5 秒后显示。</p>
</body>

</html>
```

另一个简单的计时

本例中的程序会执行 2 秒、4 秒和 6 秒的计时。

```
<html>
<head>
<script type="text/javascript">
function timedText()
{
var t1=setTimeout("document.getElementById('txt').value='2 秒'",200
var t2=setTimeout("document.getElementById('txt').value='4 秒'",400
var t3=setTimeout("document.getElementById('txt').value='6 秒'",600
}
</script>
</head>

<body>

<form>
<input type="button" value="显示计时的文本" onClick="timedText()">
<input type="text" id="txt">
</form>

<p>点击上面的按钮。输入框会显示出已经逝去的时间（2、4、6 秒）。</p>
</body>

</html>
```

在一个无穷循环中的计时事件

在本例中，单击开始计时按钮后，程序开始从 0 以秒计时。

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}
</script>
</head>

<body>

<form>
<input type="button" value="开始计时！" onClick="timedCount()">
<input type="text" id="txt">
</form>

<p>请点击上面的按钮。输入框会从 0 开始一直进行计时。</p>

</body>

</html>
```

带有停止按钮的无穷循环中的计时事件

在本例中，点击计数按钮后根据用户输入的数值开始倒计时，点击停止按钮停止计时。

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}

function stopCount()
{
c=0;
setTimeout("document.getElementById('txt').value=0",0);
clearTimeout(t);
}
</script>
</head>

<body>

<form>
<input type="button" value="开始计时！" onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="停止计时！" onClick="stopCount()">
</form>

<p>请点击上面的“开始计时”按钮来启动计时器。输入框会一直进行计时，从 0 开始。点

</body>

</html>
```

使用计时事件制作的钟表

一个 JavaScript 小时钟

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date()
var h=today.getHours()
var m=today.getMinutes()
var s=today.getSeconds()
// add a zero in front of numbers<10
m=checkTime(m)
s=checkTime(s)
document.getElementById('txt').innerHTML=h+":"+m+":"+s
t=setTimeout('startTime()',500)
}

function checkTime(i)
{
if (i<10)
  {i="0" + i}
  return i
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

JavaScript 计时事件

通过使用 JavaScript，我们有能力作到在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。我们称之为计时事件。

在 JavaScritp 中使用计时事件是很容易的，两个关键方法是:

setTimeout()

未来的某时执行代码

clearTimeout()

取消setTimeout()

setTimeout()

语法

```
var t=setTimeout("javascript语句",毫秒)
```

setTimeout() 方法会返回某个值。在上面的语句中，值被储存在名为 t 的变量中。假如你希望取消这个 setTimeout()，你可以使用这个变量名来指定它。

setTimeout() 的第一个参数是含有 JavaScript 语句的字符串。这个语句可能诸如 "alert('5 seconds!')"，或者对函数的调用，诸如 alertMsg()"。

第二个参数指示从当前起多少毫秒后执行第一个参数。

提示：1000 毫秒等于一秒。

实例

当下面这个例子中的按钮被点击时，一个提示框会在5秒中后弹出。

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
 {
 `var t=setTimeout("alert('5 seconds!')",5000)`
 }
</script>
</head>

<body>
<form>
<input type="button" value="Display timed alertbox!" onClick="timec
</form>
</body>
</html>
```

实例 - 无穷循环

要创建一个运行于无穷循环中的计时器，我们需要编写一个函数来调用其自身。在下面的例子中，当按钮被点击后，输入域便从 0 开始计数。

```
<html>

<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
 {
 document.getElementById('txt').value=c
 c=c+1
 `t=setTimeout("timedCount()",1000)`
 }
</script>
</head>

<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()">
<input type="text" id="txt">
</form>
</body>

</html>
```

clearTimeout()

语法

```
clearTimeout(setTimeout_variable)
```

实例

下面的例子和上面的无穷循环的例子相似。唯一的不同是，现在我们添加了一个 "Stop Count!" 按钮来停止这个计数器：

```
<html>

<head>
<script type="text/javascript">
var c=0
var t

function timedCount()
 {
 document.getElementById('txt').value=c
 c=c+1
 `t=setTimeout("timedCount()",1000)`
 }

function stopCount()
 {
 `clearTimeout(t)`
 }
</script>
</head>

<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="Stop count!" onClick="stopCount()">
</form>
</body>

</html>
```

JavaScript Cookies

cookie 用来识别用户。

实例

创建一个欢迎 cookie

利用用户在提示框中输入的数据创建一个 JavaScript Cookie，当该用户再次访问该页面时，根据 cookie 中的信息发出欢迎信息。

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
{
c_start=document.cookie.indexOf(c_name + "=")
if (c_start!=-1)
{
c_start=c_start + c_name.length+1
c_end=document.cookie.indexOf(";",c_start)
if (c_end==-1) c_end=document.cookie.length
return unescape(document.cookie.substring(c_start,c_end))
}
}
return ""
}

function setCookie(c_name,value,expiredays)
{
var exdate=new Date()
exdate.setDate(exdate.getDate()+expiredays)
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : "; expires="+exdate.toGMTString())
}

function checkCookie()
{
username=getCookie('username')
if (username!=null && username!="")
  {alert('Welcome again '+username+'!')}
else
  {
  username=prompt('Please enter your name:',"")
  if (username!=null && username!="")
    {
    setCookie('username',username,365)
    }
  }
}
</script>
</head>
<body onLoad="checkCookie()">
</body>
</html>
```

什么是cookie?

cookie 是存储于访问者的计算机中的变量。每当同一台计算机通过浏览器请求某个页面时，就会发送这个 cookie。你可以使用 JavaScript 来创建和取回 cookie 的值。

有关cookie的例子：

名字 cookie

当访问者首次访问页面时，他或她也许会填写他/她们的名字。名字会存储于 cookie 中。当访问者再次访问网站时，他们会收到类似 "Welcome John Doe!" 的欢迎词。而名字则是从 cookie 中取回的。

密码 cookie

当访问者首次访问页面时，他或她也许会填写他/她们的密码。密码也可被存储于 cookie 中。当他们再次访问网站时，密码就会从 cookie 中取回。

日期 cookie

当访问者首次访问你的网站时，当前的日期可存储于 cookie 中。当他们再次访问网站时，他们会收到类似这样的一条消息："Your last visit was on Tuesday August 11, 2005!"。日期也是从 cookie 中取回的。

创建和存储 cookie

在这个例子中我们要创建一个存储访问者名字的 cookie。当访问者首次访问网站时，他们会被要求填写姓名。名字会存储于 cookie 中。当访问者再次访问网站时，他们就会收到欢迎词。

首先，我们会创建一个可在 cookie 变量中存储访问者姓名的函数：

```
function setCookie(c_name,value,expiredays)
{
var exdate=new Date()
exdate.setDate(exdate.getDate()+expiredays)
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString())
}
```

上面这个函数中的参数存有 cookie 的名称、值以及过期天数。

在上面的函数中，我们首先将天数转换为有效的日期，然后，我们将 cookie 名称、值及其过期日期存入 document.cookie 对象。

之后，我们要创建另一个函数来检查是否已设置 cookie：

```
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=")
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1
    c_end=document.cookie.indexOf(";",c_start)
    if (c_end==-1) c_end=document.cookie.length
    return unescape(document.cookie.substring(c_start,c_end))
    }
  }
return ""
}
```

上面的函数首先会检查 document.cookie 对象中是否存有 cookie。假如 document.cookie 对象存有某些 cookie，那么会继续检查我们指定的 cookie 是否已储存。如果找到了我们要的 cookie，就返回值，否则返回空字符串。

最后，我们要创建一个函数，这个函数的作用是：如果 cookie 已设置，则显示欢迎词，否则显示提示框来要求用户输入名字。

```
function checkCookie()
{
username=getCookie('username')
if (username!=null && username!="")
  {alert('Welcome again '+username+'!')}
else
  {
  username=prompt('Please enter your name:',"")
  if (username!=null && username!="")
    {
    setCookie('username',username,365)
    }
  }
}
```

这是所有的代码：

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=")
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1
    c_end=document.cookie.indexOf(";",c_start)
    if (c_end==-1) c_end=document.cookie.length
    return unescape(document.cookie.substring(c_start,c_end))
    }
  }
return ""
}

function setCookie(c_name,value,expiredays)
{
var exdate=new Date()
exdate.setDate(exdate.getDate()+expiredays)
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString())
}

function checkCookie()
{
username=getCookie('username')
if (username!=null && username!="")
  {alert('Welcome again '+username+'!')}
else
  {
  username=prompt('Please enter your name:',"")
  if (username!=null && username!="")
    {
    setCookie('username',username,365)
    }
  }
}
</script>
</head>

<body onLoad="checkCookie()">
</body>
</html>
```

HTML DOM

HTML DOM 简介

HTML DOM 定义了访问和操作 **HTML** 文档的标准。

您应该具备的基础知识

在您继续学习之前，您需要对以下内容拥有基本的了解：

- HTML
- CSS
- JavaScript

如果您需要首先学习这些项目，请访问我们的首页。

什么是 DOM？

DOM 是 W3C（万维网联盟）的标准。

DOM 定义了访问 HTML 和 XML 文档的标准：

> “W3C 文档对象模型 （DOM） 是中立于平台和语言的接口，它允许程序和脚本动态地访问和更新文档的内容、结构和样式。”

W3C DOM 标准被分为 3 个不同的部分：

- 核心 DOM - 针对任何结构化文档的标准模型
- XML DOM - 针对 XML 文档的标准模型
- HTML DOM - 针对 HTML 文档的标准模型

编者注：DOM 是 Document Object Model（文档对象模型）的缩写。

什么是 XML DOM？

XML DOM 定义了所有 XML 元素的对象和属性，以及访问它们的方法。

如果您需要学习 XML DOM，请访问我们的 XML DOM 教程。

什么是 HTML DOM？

HTML DOM 是：

- HTML 的标准对象模型
- HTML 的标准编程接口
- W3C 标准

HTML DOM 定义了所有 HTML 元素的对象和属性，以及访问它们的方法。

换言之，*HTML DOM* 是关于如何获取、修改、添加或删除 *HTML* 元素的标准。

HTML DOM 节点

在 **HTML DOM** 中，所有事物都是节点。**DOM** 是被视为节点树的 **HTML**。

DOM 节点

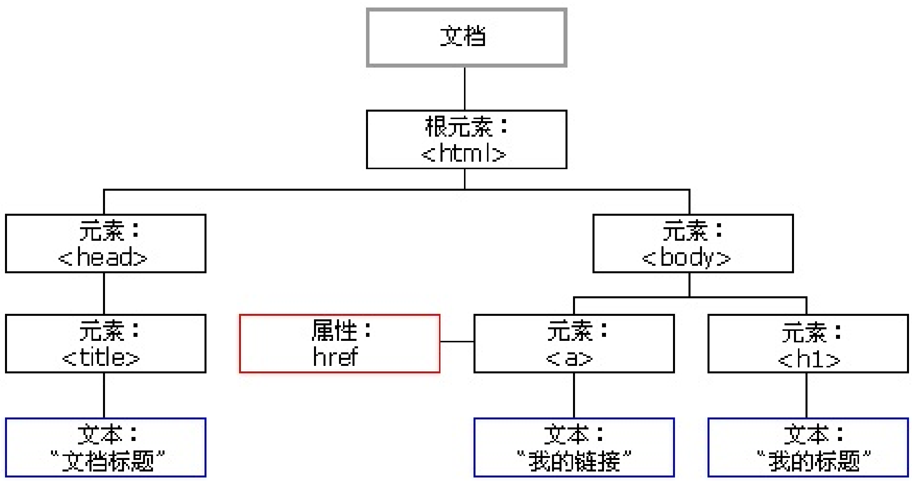
根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：

- 整个文档是一个文档节点
- 每个 HTML 元素是元素节点
- HTML 元素内的文本是文本节点
- 每个 HTML 属性是属性节点
- 注释是注释节点

HTML DOM 节点树

HTML DOM 将 HTML 文档视作树结构。这种结构被称为节点树：

HTML DOM Tree 实例



通过 HTML DOM，树中的所有节点均可通过 JavaScript 进行访问。所有 HTML 元素（节点）均可被修改，也可以创建或删除节点。

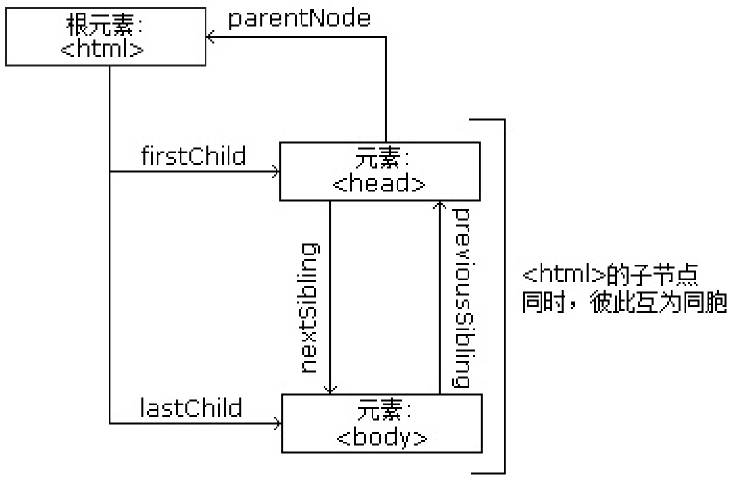
节点父、子和同胞

节点树中的节点彼此拥有层级关系。

父（parent）、子（child）和同胞（sibling）等术语用于描述这些关系。父节点拥有子节点。同级的子节点被称为同胞（兄弟或姐妹）。

- 在节点树中，顶端节点被称为根（root）
- 每个节点都有父节点、除了根（它没有父节点）
- 一个节点可拥有任意数量的子
- 同胞是拥有相同父节点的节点

下面的图片展示了节点树的一部分，以及节点之间的关系：



请看下面的 HTML 片段：

```
<html>
  <head>
    <title>DOM 教程</title>
  </head>
  <body>
    <h1>DOM 第一课</h1>
    <p>Hello world!</p>
  </body>
</html>
```

从上面的 HTML 中：

- `<html>` 节点没有父节点；它是根节点
- `<head>` 和 `<body>` 的父节点是 `<html>` 节点
- 文本节点 "Hello world!" 的父节点是 `<p>` 节点

并且：

- `<html>` 节点拥有两个子节点：`<head>` 和 `<body>`
- `<head>` 节点拥有一个子节点：`<title>` 节点
- `<title>` 节点也拥有一个子节点：文本节点 "DOM 教程"
- `<h1>` 和 `<p>` 节点是同胞节点，同时也是 `<body>` 的子节点

并且：

- `<head>` 元素是 `<html>` 元素的首个子节点

- `<body>` 元素是 `<html>` 元素的最后一个子节点
- `<h1>` 元素是 `<body>` 元素的首个子节点
- `<p>` 元素是 `<body>` 元素的最后一个子节点

警告！

DOM 处理中的常见错误是希望元素节点包含文本。

在本例中：*<title>DOM* 教程*</title>*，元素节点 <title>，包含值为 "DOM 教程" 的文本节点。

可通过节点的 *innerHTML* 属性来访问文本节点的值。

您将在稍后的章节中学习更多有关 innerHTML 属性的知识。

HTML DOM 方法

方法是我们可以在节点（HTML 元素）上执行的动作。

编程接口

可通过 JavaScript （以及其他编程语言）对 HTML DOM 进行访问。

所有 HTML 元素被定义为对象，而编程接口则是对象方法和对象属性。

方法是您能够执行的动作（比如添加或修改元素）。

属性是您能够获取或设置的值（比如节点的名称或内容）。

getElementById() 方法

getElementById() 方法返回带有指定 ID 的元素：

例子

```
var element=document.getElementById("intro");
```

HTML DOM 对象 - 方法和属性

一些常用的 HTML DOM 方法：

- getElementById(id) - 获取带有指定 id 的节点（元素）
- appendChild(node) - 插入新的子节点（元素）
- removeChild(node) - 删除子节点（元素）

一些常用的 HTML DOM 属性：

- innerHTML - 节点（元素）的文本值
- parentNode - 节点（元素）的父节点
- childNodes - 节点（元素）的子节点
- attributes - 节点（元素）的属性节点

您将在本教程的下一章中学到更多有关属性的知识。

现实生活中的对象

某个人是一个对象。

人的方法可能是 eat(), sleep(), work(), play() 等等。

所有人都有这些方法，但会在不同时间执行它们。

一个人的属性包括姓名、身高、体重、年龄、性别等等。

所有人都有这些属性，但它们的值因人而异。

一些 DOM 对象方法

这里提供一些您将在本教程中学到的常用方法：

方法	描述
getElementById()	返回带有指定 ID 的元素。
getElementsByTagName()	返回包含带有指定标签名称的所有元素的节点列表（集合/节点数组）。
getElementsByClassName()	返回包含带有指定类名的所有元素的节点列表。
appendChild()	把新的子节点添加到指定节点。
removeChild()	删除子节点。
replaceChild()	替换子节点。
insertBefore()	在指定的子节点前面插入新的子节点。
createAttribute()	创建属性节点。
createElement()	创建元素节点。
createTextNode()	创建文本节点。
getAttribute()	返回指定的属性值。
setAttribute()	把指定属性设置或修改为指定的值。

HTML DOM 属性

属性是节点（HTML 元素）的值，您能够获取或设置。

编程接口

可通过 JavaScript （以及其他编程语言）对 HTML DOM 进行访问。

所有 HTML 元素被定义为对象，而编程接口则是对象方法和对象属性。

方法是您能够执行的动作（比如添加或修改元素）。

属性是您能够获取或设置的值（比如节点的名称或内容）。

innerHTML 属性

获取元素内容的最简单方法是使用 innerHTML 属性。

innerHTML 属性对于获取或替换 HTML 元素的内容很有用。

实例

下面的代码获取 id="intro" 的 <p> 元素的 innerHTML：

实例

```
<html>
<body>

<p id="intro">Hello World!</p>

<script>
var txt=document.getElementById("intro").innerHTML;
document.write(txt);
</script>

</body>
</html>
```

在上面的例子中，getElementById 是一个方法，而 innerHTML 是属性。

innerHTML 属性可用于获取或改变任意 HTML 元素，包括 <html> 和 <body>。

nodeName 属性

nodeName 属性规定节点的名称。

- nodeName 是只读的
- 元素节点的 nodeName 与标签名相同
- 属性节点的 nodeName 与属性名相同
- 文本节点的 nodeName 始终是 #text
- 文档节点的 nodeName 始终是 #document

注释：nodeName 始终包含 HTML 元素的大写字母标签名。

nodeValue 属性

nodeValue 属性规定节点的值。

- 元素节点的 nodeValue 是 undefined 或 null
- 文本节点的 nodeValue 是文本本身
- 属性节点的 nodeValue 是属性值

获取元素的值

下面的例子会取回 <p id="intro"> 标签的文本节点值：

实例

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">
x=document.getElementById("intro");
document.write(x.firstChild.nodeValue);
</script>

</body>
</html>
```

nodeType 属性

nodeType 属性返回节点的类型。nodeType 是只读的。

比较重要的节点类型有：

元素类型	NodeType
元素	1
属性	2
文本	3
注释	8
文档	9

HTML DOM 访问

访问 HTML DOM - 查找 HTML 元素。

访问 HTML 元素（节点）

访问 HTML 元素等同于访问节点

您能够以不同的方式来访问 HTML 元素：

- 通过使用 getElementById() 方法
- 通过使用 getElementsByTagName() 方法
- 通过使用 getElementsByClassName() 方法

getElementById() 方法

getElementById() 方法返回带有指定 ID 的元素：

语法

```
node.getElementById("id");
```

下面的例子获取 id="intro" 的元素：

实例

```
document.getElementById("intro");
```

getElementsByTagName() 方法

getElementsByTagName() 返回带有指定标签名的所有元素。

语法

```
node.getElementsByTagName("tagname");
```

下面的例子返回包含文档中所有 <p> 元素的列表：

实例 1

```
document.getElementsByTagName("p");
```

下面的例子返回包含文档中所有 <p> 元素的列表，并且这些 <p> 元素应该是 id="main" 的元素的后代（子、孙等等）：

实例 2

```
document.getElementById("main").getElementsByTagName("p");
```

getElementsByClassName() 方法

如果您希望查找带有相同类名的所有 HTML 元素，请使用这个方法：

```
document.getElementsByClassName("intro");
```

上面的例子返回包含 class="intro" 的所有元素的一个列表：

注释：getElementsByClassName() 在 Internet Explorer 5,6,7,8 中无效。

HTML DOM - 修改

修改 HTML = 改变元素、属性、样式和事件。

修改 HTML 元素

修改 HTML DOM 意味着许多不同的方面：

- 改变 HTML 内容
- 改变 CSS 样式
- 改变 HTML 属性
- 创建新的 HTML 元素
- 删除已有的 HTML 元素
- 改变事件（处理程序）

在接下来的章节，我们会深入学习修改 HTML DOM 的常用方法。

创建 HTML 内容

改变元素内容的最简答的方法是使用 innerHTML 属性。

下面的例子改变一个 <p> 元素的 HTML 内容：

实例

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML="New text!";
</script>

</body>
</html>
```

提示：我们将在下面的章节为您解释例子中的细节。

改变 HTML 样式

通过 HTML DOM，您能够访问 HTML 元素的样式对象。

下面的例子改变一个段落的 HTML 样式：

实例

```
<html>

<body>
<p id="p2">Hello world!</p>

<script>
document.getElementById("p2").style.color="blue";
</script>

</body>
</html>
```

创建新的 HTML 元素

如需向 HTML DOM 添加新元素，您首先必须创建该元素（元素节点），然后把它追加到已有的元素上。

实例

```
<div id="d1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new.");
para.appendChild(node);

var element=document.getElementById("d1");
element.appendChild(para);
</script>
```

HTML DOM - 修改 HTML 内容

通过 HTML DOM，JavaScript 能够访问 HTML 文档中的每个元素。

改变 HTML 内容

改变元素内容的最简答的方法是使用 innerHTML 属性。

下面的例子更改 <p> 元素的 HTML 内容：

实例

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML="New text!";
</script>

</body>
</html>
```

改变 HTML 样式

通过 HTML DOM，您能够访问 HTML 对象的样式对象。

下面的例子更改段落的 HTML 样式：

实例

```
<html>

<body>
<p id="p2">Hello world!</p>

<script>
document.getElementById("p2").style.color="blue";
</script>

</body>
</html>
```

使用事件

HTML DOM 允许您在事件发生时执行代码。

当 HTML 元素”有事情发生“时，浏览器就会生成事件：

- 在元素上点击
- 加载页面
- 改变输入字段

你可以在下一章学习更多有关事件的内容。

下面两个例子在按钮被点击时改变 <body> 元素的背景色：

实例

```
<html>
<body>

<input type="button" onclick="document.body.style.backgroundColor=
value="Change background color" />

</body>
</html>
```

在本例中，由函数执行相同的代码：

实例

```
<html>
<body>

<script>
function ChangeBackground()
{
document.body.style.backgroundColor="lavender";
}
</script>

<input type="button" onclick="ChangeBackground()"
value="Change background color" />

</body>
</html>
```

下面的例子在按钮被点击时改变 <p> 元素的文本：

实例

```
<html>
<body>

<p id="p1">Hello world!</p>

<script>
function ChangeText()
{
document.getElementById("p1").innerHTML="New text!";
}
</script>

<input type="button" onclick="ChangeText()" value="Change text">

</body>
</html>
```

HTML DOM - 元素

添加、删除和替换 HTML 元素。

创建新的 HTML 元素 - appendChild()

如需向 HTML DOM 添加新元素，您首先必须创建该元素，然后把它追加到已有的元素上。

实例

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new.");
para.appendChild(node);

var element=document.getElementById("div1");
element.appendChild(para);
</script>
```

例子解释

这段代码创建了一个新的 <p> 元素：

```
var para=document.createElement("p");
```

如需向 <p> 元素添加文本，您首先必须创建文本节点。这段代码创建文本节点：

```
var node=document.createTextNode("This is a new paragraph.");
```

然后您必须向 <p> 元素追加文本节点：

```
para.appendChild(node);
```

最后，您必须向已有元素追加这个新元素。

这段代码查找到一个已有的元素：

```
var element=document.getElementById("div1");
```

这段代码向这个已存在的元素追加新元素：

```
element.appendChild(para);
```

创建新的 HTML 元素 - insertBefore()

上一个例子中的 appendChild() 方法，将新元素作为父元素的最后一个子元素进行添加。

如果不希望如此，您可以使用 insertBefore() 方法：

实例

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new.");
para.appendChild(node);

var element=document.getElementById("div1");
var child=document.getElementById("p1");
element.insertBefore(para,child);
</script>
```

删除已有的 HTML 元素

如需删除 HTML 元素，您必须清楚该元素的父元素：

实例

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
<script>
var parent=document.getElementById("div1");
var child=document.getElementById("p1");
parent.removeChild(child);
</script>
```

例子解释

这个 HTML 文档包含一个带有两个子节点（两个 <p> 元素）的 <div> 元素：

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
```

查找 id="div1" 的元素：

```
var parent=document.getElementById("div1");
```

查找 id="p1" 的 <p> 元素：

```
var child=document.getElementById("p1");
```

从父元素中删除子元素：

```
parent.removeChild(child);
```

提示：能否在不引用父元素的情况下删除某个元素？

很抱歉。DOM 需要了解您需要删除的元素，以及它的父元素。

这里提供一个常用的解决方法：找到您需要删除的子元素，然后使用 parentNode 属性来查找其父元素：

```
var child=document.getElementById("p1");
child.parentNode.removeChild(child);
```

替换 HTML 元素

如需替换 HTML DOM 中的元素，请使用 replaceChild() 方法：

实例

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new.");
para.appendChild(node);

var parent=document.getElementById("div1");
var child=document.getElementById("p1");
parent.replaceChild(para,child);
</script>
```

HTML DOM - 事件

HTML DOM 允许 JavaScript 对 HTML 事件作出反应。。

实例

```
<div style="background-color:#8AB83D;width:160px;height:60px;
margin:20px;padding-top:20px;color:#ffffff;font-weight:bold;
font-size:14px;float:left;text-align:center;"
onmouseover="this.innerHTML='Thank You'"
onmouseout="this.innerHTML='Mouse Over Me'">Mouse Over Me</div>

<div style="background-color:#8AB83D;width:160px;height:60px;
margin:20px;padding-top:20px;color:#ffffff;font-weight:bold;
font-size:14px;float:left;text-align:center;"
onclick="clickMeEvent(this)">Click Me</div>

<script type="application/javascript">
    <!--
    function clickMeEvent(obj)
    {
    if (obj.innerHTML=="Goodbye")
        {
        obj.style.display="none";
        }
    else if (obj.innerHTML=="Thank You")
        {
        obj.innerHTML="Goodbye";
        }
    else if (obj.innerHTML=="Click Me<br>Click Me Again<br>And Aga
        {
        obj.innerHTML="Thank You";
        }
    else if (obj.innerHTML=="Click Me<br>Click Me Again")
        {
        obj.innerHTML=obj.innerHTML+"<br>And Again";
        }
    else
        {
        obj.innerHTML=obj.innerHTML+"<br>Click Me Again";
        }
    }
    //-->
</script>
```

对事件作出反应

当事件发生时，可以执行 JavaScript，比如当用户点击一个 HTML 元素时。

如需在用户点击某个元素时执行代码，请把 JavaScript 代码添加到 HTML 事件属性中：

```
onclick=_JavaScript_
```

HTML 事件的例子：

- 当用户点击鼠标时
- 当网页已加载时
- 当图片已加载时
- 当鼠标移动到元素上时
- 当输入字段被改变时
- 当 HTML 表单被提交时
- 当用户触发按键时

在本例中，当用户点击时，会改变 <h1> 元素的内容：

实例

```
<!DOCTYPE html>
<html>
<body>
<h1 onclick="this.innerHTML='hello!'">请点击这段文本!</h1>
</body>
</html>
```

在本例中，会从事件处理程序中调用函数：

实例

```
<!DOCTYPE html>
<html>
<head>
<script>
function changetext(id)
{
id.innerHTML="hello!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">请点击这段文本!</h1>
</body>
</html>
```

HTML 事件属性

如需向 HTML 元素分配事件，您可以使用事件属性。

实例

向 button 元素分配一个 onclick 事件：

```
<button onclick="displayDate()">试一试</button>
```

在上面的例子中，当点击按钮时，会执行名为 displayDate 的函数。

使用 HTML DOM 来分配事件

HTML DOM 允许您使用 JavaScript 向 HTML 元素分配事件：

实例

为 button 元素分配 onclick 事件：

```
<script>
document.getElementById("myBtn").onclick=function(){displayDate()};
</script>
```

在上面的例子中，名为 displayDate 的函数被分配给了 id=myButn" 的 HTML 元素。

当按钮被点击时，将执行函数。

onload 和 onunload 事件

当用户进入或离开页面时，会触发 onload 和 onunload 事件。

onload 事件可用于检查访客的浏览器类型和版本，以便基于这些信息来加载不同版本的网页。

onload 和 onunload 事件可用于处理 cookies。

实例

```
<body onload="checkCookies()">
```

onchange 事件

onchange 事件常用于输入字段的验证。

下面的例子展示了如何使用 onchange。当用户改变输入字段的内容时，将调用 upperCase() 函数。

实例

```
<input type="text" id="fname" onchange="upperCase()">
```

onmouseover 和 onmouseout 事件

onmouseover 和 onmouseout 事件可用于在鼠标指针移动到或离开元素时触发函数。

实例

一个简单的 onmouseover-onmouseout 例子：

```
<div style="background-color:#8AB83D;width:150px;height:60px;
margin:20px;padding-top:20px;color:#ffffff;font-weight:bold;
font-size:18px;text-align:center;" onmouseover="this.innerHTML='谢谢

<p class="tiy"><a target="_blank" href="/tiy/t.asp?f=dom_event_onmo
</div>
```

onmousedown、onmouseup 以及 onclick 事件

onmousedown、onmouseup 以及 onclick 事件是鼠标点击的全部过程。首先当某个鼠标按钮被点击时，触发 onmousedown 事件，然后，当鼠标按钮被松开时，会触发 onmouseup 事件，最后，当鼠标点击完成时，触发 onclick 事件。

实例

一个简单的 onmousedown-onmouseup 实例：

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)" style="backgro
margin:20px;padding-top:20px;color:#ffffff;font-weight:bold;
font-size:18px;text-align:center;">点击这里</div>

<script type="application/javascript">
    <!--
    function mDown(obj)
    {
    obj.style.backgroundColor="#1ec5e5";
    obj.innerHTML="松开鼠标"
    }

    function mUp(obj)
    {
    obj.style.backgroundColor="#8AB83D";
    obj.innerHTML="谢谢"
    }

    //-->
</script>
```

HTML DOM 事件对象参考手册

如需每个事件的完整描述和例子，请访问我们的 HTML DOM 参考手册。

HTML DOM - 导航

通过 HTML DOM，您能够使用节点关系在节点树中导航。

HTML DOM 节点列表

getElementsByTagName() 方法返回节点列表。节点列表是一个节点数组。

下面的代码选取文档中的所有 <p> 节点：

实例

```
var x=document.getElementsByTagName("p");
```

可以通过下标号访问这些节点。如需访问第二个 <p>，您可以这么写：

```
y=x[1];
```

注释：下标号从 0 开始。

HTML DOM 节点列表长度

length 属性定义节点列表中节点的数量。

您可以使用 length 属性来循环节点列表：

实例

```
x=document.getElementsByTagName("p");

for (i=0;i<x.length;i++)
{
document.write(x[i].innerHTML);
document.write("<br />");
}
```

例子解释：

- 获取所有 `<p>` 元素节点

- 输出每个 `<p>` 元素的文本节点的值

导航节点关系

您能够使用三个节点属性：parentNode、firstChild 以及 lastChild ，在文档结构中进行导航。

请看下面的 HTML 片段：

```
<html>
<body>

<p>Hello World!</p>
<div>
  <p>DOM 很有用!</p>
  <p>本例演示节点关系。</p>
</div>

</body>
</html>
```

- 首个 `<p>` 元素是 `<body>` 元素的首个子元素（firstChild）
- `<div>` 元素是 `<body>` 元素的最后一个子元素（lastChild）
- `<body>` 元素是首个 `<p>` 元素和 `<div>` 元素的父节点（parentNode）

firstChild 属性可用于访问元素的文本：

实例

```
<html>
<body>

<p id="intro">Hello World!</p>

<script>
x=document.getElementById("intro");
document.write(x.firstChild.nodeValue);
</script>

</body>
</html>
```

DOM 根节点

这里有两个特殊的属性，可以访问全部文档：

- document.documentElement - 全部文档
- document.body - 文档的主体

实例

```
<html>
<body>

<p>Hello World!</p>
<div>
<p>DOM 很有用!</p>
<p>本例演示 <b>document.body</b> 属性。</p>
</div>

<script>
alert(document.body.innerHTML);
</script>

</body>
</html>
```

childNodes 和 nodeValue

除了 innerHTML 属性，您也可以使用 childNodes 和 nodeValue 属性来获取元素的内容。

下面的代码获取 id="intro" 的 <p> 元素的值：

实例

```
<html>
<body>

<p id="intro">Hello World!</p>

<script>
var txt=document.getElementById("intro").childNodes[0].nodeValue;
document.write(txt);
</script>

</body>
</html>
```

在上面的例子中，getElementById 是一个方法，而 childNodes 和 nodeValue 是属性。

在本教程中，我们将使用 innerHTML 属性。不过，学习上面的方法有助于对 DOM 树结构和导航的理解。

JavaScript HTML DOM EventListener

addEventListener() 方法

实例

点用户点击按钮时触发监听事件：

```
document.getElementById("myBtn").addEventListener("click", displayD
```

addEventListener() 方法用于向指定元素添加事件句柄。

addEventListener() 方法添加的事件句柄不会覆盖已存在的事件句柄。

你可以向一个元素添加多个事件句柄。

你可以向同个元素添加多个同类型的事件句柄，如：两个 "click" 事件。

你可以向任何 DOM 对象添加事件监听，不仅仅是 HTML 元素。如： window 对象。

addEventListener() 方法可以更简单的控制事件（冒泡与捕获）。

当你使用 addEventListener() 方法时, JavaScript 从 HTML 标记中分离开来，可读性更强， 在没有控制HTML标记时也可以添加事件监听。

你可以使用 removeEventListener() 方法来移除事件的监听。

语法

element.addEventListener(*event, function, useCapture*);

第一个参数是事件的类型 (如 "click" 或 "mousedown").

第二个参数是事件触发后调用的函数。

第三个参数是个布尔值用于描述事件是冒泡还是捕获。该参数是可选的。

> 注意:不要使用 "on" 前缀。 例如，使用 "click" ,而不是使用 "onclick"。

向原元素添加事件句柄

实例

当用户点击元素时弹出 "Hello World!" :

```
_element_.addEventListener("click", function(){ alert("Hello World
```

你可以使用函数名，来引用外部函数:

实例

当用户点击元素时弹出 "Hello World!" :

```
_element_.addEventListener("click", myFunction);

function myFunction() {
    alert ("Hello World!");
}
```

向同一个元素中添加多个事件句柄

addEventListener() 方法允许向同个元素添加多个事件，且不会覆盖已存在的事件：

实例

```
_element_.addEventListener("click", myFunction);
_element_.addEventListener("click", mySecondFunction);
```

你可以向同个元素添加不同类型的事件：

实例

```
_element_.addEventListener("mouseover", myFunction);
_element_.addEventListener("click", mySecondFunction);
_element_.addEventListener("mouseout", myThirdFunction);
```

向 Window 对象添加事件句柄

addEventListener() 方法允许你在 HTML DOM 对象添加事件监听， HTML DOM 对象如： HTML 元素, HTML 文档, window 对象。或者其他支出的事件对象如: xmlHttpRequest 对象。

实例

当用户重置窗口大小时添加事件监听：

```
window.addEventListener("resize", function(){
    document.getElementById("demo").innerHTML = _sometext_;
});
```

传递参数

当传递参数值时，使用"匿名函数"调用带参数的函数：

实例

```
_element_.addEventListener("click", function(){ myFunction(p1, p2);
```

事件冒泡或事件捕获？

事件传递有两种方式：冒泡与捕获。

事件传递定义了元素事件触发的顺序。 如果你将 <p> 元素插入到 <div> 元素中，用户点击 <p> 元素, 哪个元素的 "click" 事件先被触发呢？

在 冒泡 中，内部元素的事件会先被触发，然后再触发外部元素，即： <p> 元素的点击事件先触发，然后会触发 <div> 元素的点击事件。

在 捕获 中，外部元素的事件会先被触发，然后才会触发内部元素的事件，即： <div> 元素的点击事件先触发 ，然后再触发 <p> 元素的点击事件。

addEventListener() 方法可以指定 "useCapture" 参数来设置传递类型：

addEventListener(*event*, *function*, *useCapture*);

默认值为 false, 及冒泡传递，当值为 true 时, 事件使用捕获传递。

实例

```
document.getElementById("myDiv").addEventListener("click", myFuncti
```

removeEventListener() 方法

removeEventListener() 方法移除由 addEventListener() 方法添加的事件句柄:

实例

```
_element_.removeEventListener("mousemove", myFunction);
```

浏览器支持

表格中的数字表示支持该方法的第一个浏览器的版本号。

方法					
addEventListener()	1.0	9.0	1.0	1.0	7.0
removeEventListener()	1.0	9.0	1.0	1.0	7.0

注意： IE 8 及更早 IE 版本，Opera 7.0及其更早版本不支持 addEventListener() 和 removeEventListener() 方法。但是，对于这类浏览器版本可以使用 detachEvent() 方法来移除事件句柄:

element.attachEvent(event, function);
element.detachEvent(event, function);

实例

跨浏览器解决方法:

```
var x = document.getElementById("myBtn");
if (x.addEventListener) {                    // 所有主流浏览器，除了 I
    x.addEventListener("click", myFunction);
} else if (x.attachEvent) {                  // IE 8 及更早版本
    x.attachEvent("onclick", myFunction);
}
```

HTML DOM 事件对象参考手册

所有 HTML DOM 事件，可以查看我们完整的 HTML DOM Event 对象参考手册。

AJAX 教程

AJAX 简介

AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

您应当具备的基础知识

在继续学习之前，您需要对下面的知识有基本的了解：

- HTML / XHTML
- CSS
- JavaScript / DOM

如果您希望首先学习这些项目，请在我们的首页访问这些教程。

什么是 AJAX ？

AJAX = 异步 JavaScript 和 XML。

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、开心网等等。

Google Suggest

在 2005 年，Google 通过其 Google Suggest 使 AJAX 变得流行起来。

Google Suggest 使用 AJAX 创造出动态性极强的 web 界面：当您在谷歌的搜索框输入关键字时，JavaScript 会把这些字符发送到服务器，然后服务器会返回一个搜索建议的列表。

今天就开始使用 AJAX

AJAX 基于已有的标准。这些标准已被大多数开发者使用多年。

请阅读下一章，看看 AJAX 是如何工作的！

AJAX 实例

- XHR 创建对象

为了帮助您理解 **AJAX** 的工作原理，我们创建了一个小型的 **AJAX** 应用程序。

实例

```
<div id="myDiv"><p>AJAX is not a programming language.</p>
<p>It is just a technique for creating better and more interactive
<button type="button" onclick="loadXMLDoc()">通过 AJAX 改变内容</butt
```

AJAX 实例解释

上面的 AJAX 应用程序包含一个 div 和一个按钮。

div 部分用于显示来自服务器的信息。当按钮被点击时，它负责调用名为 loadXMLDoc() 的函数：

```
<html>
<body>

<div id="myDiv"><h3>Let AJAX change this text</h3></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button

</body>
</html>
```

接下来，在页面的 head 部分添加一个 <script> 标签。该标签中包含了这个 loadXMLDoc() 函数：

```
<head>
<script type="text/javascript">
function loadXMLDoc()
{
.... AJAX script goes here ...
}
</script>
</head>
```

下面的章节会为您讲解 AJAX 的工作原理。

- XHR 创建对象

AJAX - 创建 XMLHttpRequest 对象

- XHR 请求

XMLHttpRequest 是 AJAX 的基础。

XMLHttpRequest 对象

所有现代浏览器均支持 XMLHttpRequest 对象（IE5 和 IE6 使用 ActiveXObject）。

XMLHttpRequest 用于在后台与服务器交换数据。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

创建 XMLHttpRequest 对象

所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera）均内建 XMLHttpRequest 对象。

创建 XMLHttpRequest 对象的语法：

```
variable=new XMLHttpRequest();
```

老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

```
variable=new ActiveXObject("Microsoft.XMLHTTP");
```

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject ：

```
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
```

在下一章中，您将学习发送服务器请求的知识。

- XHR 请求

AJAX - 向服务器发送请求

- XHR 创建对象
- XHR 响应

XMLHttpRequest 对象用于和服务器交换数据。

向服务器发送请求

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

```
xmlhttp.open("GET","test1.txt",true);
xmlhttp.send();
```

方法	描述
open(*method*,*url*,*async*)	规定请求的类型、URL 以及是否异步处理请求。 *method*：请求的类型；GET 或 POST *url*：文件在服务器上的位置 *async*：true（异步）或 false（同步）
send(*string*)	将请求发送到服务器。 *string*：仅用于 POST 请求

GET 还是 POST？

与 POST 相比，GET 更简单也更快，并且在大部分情况下都能用。

然而，在以下情况中，请使用 POST 请求：

- 无法使用缓存文件（更新服务器上的文件或数据库）
- 向服务器发送大量数据（POST 没有数据量限制）
- 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

GET 请求

一个简单的 GET 请求：

```
xmlhttp.open("GET","demo_get.asp",true);
xmlhttp.send();
```

在上面的例子中，您可能得到的是缓存的结果。

为了避免这种情况，请向 URL 添加一个唯一的 ID：

```
xmlhttp.open("GET","demo_get.asp?t=" + `Math.random()`,true);
xmlhttp.send();
```

如果您希望通过 GET 方法发送信息，请向 URL 添加信息：

```
xmlhttp.open("GET","demo_get2.asp?fname=Bill&lname=Gates",true);
xmlhttp.send();
```

POST 请求

一个简单 POST 请求：

```
xmlhttp.open("POST","demo_post.asp",true);
xmlhttp.send();
```

如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定您希望发送的数据：

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-url
xmlhttp.send("fname=Bill&lname=Gates");
```

方法	描述
setRequestHeader(*header*,*value*)	向请求添加 HTTP 头。 *header*: 规定头的名称 *value*: 规定头的值

url - 服务器上的文件

open() 方法的 *url* 参数是服务器上文件的地址：

```
xmlhttp.open("GET","ajax_test.asp",true);
```

该文件可以是任何类型的文件，比如 .txt 和 .xml，或者服务器脚本文件，比如 .asp 和 .php （在传回响应之前，能够在服务器上执行任务）。

异步 - True 或 False？

AJAX 指的是异步 JavaScript 和 XML（Asynchronous JavaScript and XML）。

XMLHttpRequest 对象如果要用于 AJAX 的话，其 open() 方法的 async 参数必须设置为 true：

```
xmlhttp.open("GET","ajax_test.asp",`true`);
```

对于 web 开发人员来说，发送异步请求是一个巨大的进步。很多在服务器执行的任务都相当费时。AJAX 出现之前，这可能会引起应用程序挂起或停止。

通过 AJAX，JavaScript 无需等待服务器的响应，而是：

- 在等待服务器响应时执行其他脚本
- 当响应就绪后对响应进行处理

Async = true

当使用 async=true 时，请规定在响应处于 onreadystatechange 事件中的就绪状态时执行的函数：

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText
    }
  }
xmlhttp.open("GET","test1.txt",true);
xmlhttp.send();
```

您将在稍后的章节学习更多有关 onreadystatechange 的内容。

Async = false

如需使用 async=false，请将 open() 方法中的第三个参数改为 false：

```
xmlhttp.open("GET","test1.txt",false);
```

我们不推荐使用 async=false，但是对于一些小型的请求，也是可以的。

请记住，JavaScript 会等到服务器响应就绪才继续执行。如果服务器繁忙或缓慢，应用程序会挂起或停止。

注释：当您使用 async=false 时，请不要编写 onreadystatechange 函数 - 把代码放到 send() 语句后面即可：

```
xmlhttp.open("GET","test1.txt",false);
xmlhttp.send();
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

- XHR 创建对象
- XHR 响应

AJAX - 服务器响应

- XHR 请求
- XHR readyState

服务器响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

属性	描述
responseText	获得字符串形式的响应数据。
responseXML	获得 XML 形式的响应数据。

responseText 属性

如果来自服务器的响应并非 XML，请使用 responseText 属性。

responseText 属性返回字符串形式的响应，因此您可以这样使用：

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

responseXML 属性

如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，请使用 responseXML 属性：

请求 books.xml 文件，并解析响应：

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)
  {
  txt=txt + x[i].childNodes[0].nodeValue + "<br />";
  }
document.getElementById("myDiv").innerHTML=txt;
```

- XHR 请求
- XHR readyState

AJAX - onreadystatechange 事件

- XHR 响应

onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

属性	描述
onreadystatechange	存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。
readyState	存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。 0: 请求未初始化 1: 服务器连接已建立 2: 请求已接收 3: 请求处理中 4: 请求已完成，且响应已就绪
status	200: "OK" 404: 未找到页面

在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText
    }
  }
```

注释：onreadystatechange 事件被触发 5 次（0 - 4），对应着 readyState 的每个变化。

使用 Callback 函数

callback 函数是一种以参数形式传递给另一个函数的函数。

如果您的网站上存在多个 AJAX 任务，那么您应该为创建 XMLHttpRequest 对象编写一个标准的函数，并为每个 AJAX 任务调用该函数。

该函数调用应该包含 URL 以及发生 onreadystatechange 事件时执行的任务（每次调用可能不尽相同）：

```
function myFunction()
{
loadXMLDoc("ajax_info.txt",function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText
    }
  });
}
```

- XHR 响应

AJAX ASP/PHP 请求实例

- XHR readyState

AJAX 用于创造动态性更**强**的应用程序。

AJAX ASP/PHP 实例

下面的例子将为您演示当用户在输入框中键入字符时，网页如何与 web 服务器进行通信：

```
<p>请在下面的输入框中键入字母（A - Z）：</p>

<form action="" style="margin-top:15px;">
姓氏：<input id="txt1" onkeyup="showHint(this.value)" type="text">
</form>

<p>建议：<span id="txtHint"></span></p>
```

实例解释 - showHint() 函数

当用户在上面的输入框中键入字符时，会执行函数 "showHint()" 。该函数由 "onkeyup" 事件触发：

```
function showHint(str)
{
var xmlhttp;
if (str.length==0)
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseTe
    }
  }
xmlhttp.open("GET","gethint.asp?q="+str,true);
xmlhttp.send();
}
```

源代码解释：

如果输入框为空 (str.length==0)，则该函数清空 txtHint 占位符的内容，并退出函数。

如果输入框不为空，showHint() 函数执行以下任务：

- 创建 XMLHttpRequest 对象
- 当服务器响应就绪时执行函数
- 把请求发送到服务器上的文件
- 请注意我们向 URL 添加了一个参数 q （带有输入框的内容）

AJAX 服务器页面 - ASP 和 PHP

由上面的 JavaScript 调用的服务器页面是 ASP 文件，名为 "gethint.asp"。

下面，我们创建了两个版本的服务器文件，一个用 ASP 编写，另一个用 PHP 编写。

ASP 文件

"gethint.asp" 中的源代码会检查一个名字数组，然后向浏览器返回相应的名字：

```
<%
response.expires=-1
dim a(30)
'用名字来填充数组
a(1)="Anna"
a(2)="Brittany"
a(3)="Cinderella"
a(4)="Diana"
a(5)="Eva"
a(6)="Fiona"
a(7)="Gunda"
a(8)="Hege"
a(9)="Inga"
a(10)="Johanna"
a(11)="Kitty"
a(12)="Linda"
a(13)="Nina"
a(14)="Ophelia"
a(15)="Petunia"
a(16)="Amanda"
a(17)="Raquel"
a(18)="Cindy"
a(19)="Doris"
a(20)="Eve"
a(21)="Evita"
a(22)="Sunniva"
a(23)="Tove"
a(24)="Unni"
a(25)="Violet"
a(26)="Liza"
a(27)="Elizabeth"
a(28)="Ellen"
a(29)="Wenche"
a(30)="Vicky"

'获得来自 URL 的 q 参数
q=ucase(request.querystring("q"))

'如果 q 大于 0，则查找数组中的所有提示
if len(q)>0 then
  hint=""
  for i=1 to 30
    if q=ucase(mid(a(i),1,len(q))) then
      if hint="" then
        hint=a(i)
      else
        hint=hint & " , " & a(i)
```

```
      end if
    end if
  next
end if

'如果未找到提示，则输出 "no suggestion"
'否则输出正确的值
if hint="" then
  response.write("no suggestion")
else
  response.write(hint)
end if
%>
```

PHP 文件

下面的代码用 PHP 编写，与上面的 ASP 代码作用是一样的。

注释：如需在 PHP 中运行这个例子，请将 url 变量的值（Javascript 代码中）由 "gethint.asp" 改为 "gethint.php"。

```
<?php
// 用名字来填充数组
$a[]="Anna";
$a[]="Brittany";
$a[]="Cinderella";
$a[]="Diana";
$a[]="Eva";
$a[]="Fiona";
$a[]="Gunda";
$a[]="Hege";
$a[]="Inga";
$a[]="Johanna";
$a[]="Kitty";
$a[]="Linda";
$a[]="Nina";
$a[]="Ophelia";
$a[]="Petunia";
$a[]="Amanda";
$a[]="Raquel";
$a[]="Cindy";
$a[]="Doris";
$a[]="Eve";
$a[]="Evita";
$a[]="Sunniva";
$a[]="Tove";
$a[]="Unni";
$a[]="Violet";
$a[]="Liza";
$a[]="Elizabeth";
```

```
$a[]="Ellen";
$a[]="Wenche";
$a[]="Vicky";

//获得来自 URL 的 q 参数
$q=$_GET["q"];

//如果 q 大于 0，则查找数组中的所有提示
if (strlen($q) > 0)
  {
  $hint="";
  for($i=0; $i<count($a); $i++)
    {
    if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q))))
      {
      if ($hint=="")
        {
        $hint=$a[$i];
        }
      else
        {
        $hint=$hint." , ".$a[$i];
        }
      }
    }
  }

// 如果未找到提示，则把输出设置为 "no suggestion"
// 否则设置为正确的值
if ($hint == "")
  {
  $response="no suggestion";
  }
else
  {
  $response=$hint;
  }

//输出响应
echo $response;
?>
```

- XHR readyState

AJAX 数据库实例

AJAX 可用来与数据库进行动态通信。

AJAX 数据库实例

下面的例子将演示网页如何通过 AJAX 从数据库读取信息：

```
<p>请在下面的下拉列表中选择一个客户：</p>

<form action="" style="margin-top:15px;">
<label>请选择一位客户：
<select name="customers" onchange="showCustomer(this.value)" style=
<option value="APPLE">Apple Computer, Inc.</option>
<option value="BAIDU ">BAIDU, Inc</option>
<option value="Canon">Canon USA, Inc.</option>
<option value="Google">Google, Inc.</option>
<option value="Nokia">Nokia Corporation</option>
<option value="SONY">Sony Corporation of America</option>
</select>
</label>
</form>

<div id="txtHint" style="margin:15px 0 0 0;padding:0;border:0;color
```

实例解释 - showCustomer() 函数

当用户在上面的下拉列表中选择某个客户时，会执行名为 "showCustomer()" 的函数。该函数由 "onchange" 事件触发：

```
function showCustomer(str)
{
var xmlhttp;
if (str=="")
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseTe
    }
  }
xmlhttp.open("GET","getcustomer.asp?q="+str,true);
xmlhttp.send();
}
```

showCustomer() 函数执行以下任务：

- 检查是否已选择某个客户
- 创建 XMLHttpRequest 对象
- 当服务器响应就绪时执行所创建的函数
- 把请求发送到服务器上的文件
- 请注意我们向 URL 添加了一个参数 q （带有输入域中的内容）

AJAX 服务器页面

由上面的 JavaScript 调用的服务器页面是 ASP 文件，名为 "getcustomer.asp"。

用 PHP 编写服务器文件也很容易，或者用其他服务器语言。请看用 PHP 编写的相应的例子。

"getcustomer.asp" 中的源代码负责对数据库进行查询，然后用 HTML 表格返回结果：

```
<%
response.expires=-1
sql="SELECT * FROM CUSTOMERS WHERE CUSTOMERID="
sql=sql & "'" & request.querystring("q") & "'"

set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs=Server.CreateObject("ADODB.recordset")
rs.Open sql,conn

response.write("<table>")
do until rs.EOF
  for each x in rs.Fields
    response.write("<tr><td><b>" & x.name & "</b></td>")
    response.write("<td>" & x.value & "</td></tr>")
  next
  rs.MoveNext
loop
response.write("</table>")
%>
```

AJAX XML 实例

AJAX 可用来与 **XML** 文件进行交互式通信。

AJAX XML 实例

下面的例子将演示网页如何使用 AJAX 来读取来自 XML 文件的信息：

```
<div id="txtCDInfo" style="margin:15px 0 0 0; border:0; padding:0;'
<button onclick="loadXMLDoc('/example/xmle/cd_catalog.xml')" style=
</div>
```

实例解释 - loadXMLDoc() 函数

当用户点击上面的"获得 CD 信息"这个按钮，就会执行 loadXMLDoc() 函数。

loadXMLDoc() 函数创建 XMLHttpRequest 对象，添加当服务器响应就绪时执行的函数，并将请求发送到服务器。

当服务器响应就绪时，会构建一个 HTML 表格，从 XML 文件中提取节点（元素），最后使用已经填充了 XML 数据的 HTML 表格来更新 txtCDInfo 占位符：

```
function loadXMLDoc(url)
{
var xmlhttp;
var txt,xx,x,i;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    txt="<table border='1'><tr><th>Title</th><th>Artist</th></tr>";
    x=xmlhttp.responseXML.documentElement.getElementsByTagName("CD'
    for (i=0;i<x.length;i++)
      {
      txt=txt + "<tr>";
      xx=x[i].getElementsByTagName("TITLE");
```

```
          {
          try
            {
            txt=txt + "<td>" + xx[0].firstChild.nodeValue + "</td>";
            }
          catch (er)
            {
            txt=txt + "<td> </td>";
            }
          }
      xx=x[i].getElementsByTagName("ARTIST");
        {
          try
            {
            txt=txt + "<td>" + xx[0].firstChild.nodeValue + "</td>";
            }
          catch (er)
            {
            txt=txt + "<td> </td>";
            }
          }
        txt=txt + "</tr>";
        }
      txt=txt + "</table>";
      document.getElementById('txtCDInfo').innerHTML=txt;
      }
    }
  xmlhttp.open("GET",url,true);
  xmlhttp.send();
  }
```

AJAX 服务器页面

上面这个例子中使用的服务器页面实际上是一个 XML 文件，名为 "cd_catalog.xml"。

W3School JavaScript 高级教程

来源：JavaScript高级教程

整理：飞龙

JavaScript 高级教程

JavaScript 的历史

为了发挥 **JavaScript** 的全部潜力，了解它的本质、历史及局限性是十分重要的。

本节为您讲解 **JavaScript** 和客户端脚本的起源。

Nombas 和 ScriptEase

大概在 1992 年，一家称作 Nombas 的公司开发了一种叫做 C 减减（C-minus-minus，简称 Cmm）的嵌入式脚本语言。Cmm 背后的理念很简单：一个足够强大可以替代宏操作（macro）的脚本语言，同时保持与 C （和 C ++）足够的相似性，以便开发人员能很快学会。这个脚本语言捆绑在一个叫做 CEnvi 的共享软件中，它首次向开发人员展示了这种语言的威力。

Nombas 最终把 Cmm 的名字改成了 ScriptEase，原因是后面的部分（mm）听起来过于消极，同时字母 C "令人害怕"。

现在 ScriptEase 已经成为了 Nombas 产品背后的主要驱动力。

Netscape 发明了 JavaScript

当 Netscape Navigator 崭露头角时，Nombas 开发了一个可以嵌入网页中的 CEnvi 的版本。这些早期的试验被称为 Espresso Page（浓咖啡般的页面），它们代表了第一个在万维网上使用的客户端语言。而 Nombas 丝毫没有料到它的理念将会成为万维网的一块重要基石。

当网上冲浪越来越流行时，对于开发客户端脚本的需求也逐渐增大。此时，大部分因特网用户还仅仅通过 28.8 kbit/s 的调制解调器连接到网络，即便这时网页已经不断地变得更大和更复杂。而更加加剧用户痛苦的是，仅仅为了简单的表单有效性验证，就要与服务器进行多次地往返交互。设想一下，用户填完一个表单，点击提交按钮，等待了 30 秒的处理后，看到的却是一条告诉你忘记填写一个必要的字段。

那时正处于技术革新最前沿的 Netscape，开始认真考虑开发一种客户端脚本语言来解决简单的处理问题。

当时工作于 Netscape 的 Brendan Eich，开始着手为即将在 1995 年发行的 Netscape Navigator 2.0 开发一个称之为 LiveScript 的脚本语言，当时的目的是在浏览器和服务器（本来要叫它 LiveWire）端使用它。Netscape 与 Sun 及时完成 LiveScript 实现。

就在 Netscape Navigator 2.0 即将正式发布前，Netscape 将其更名为 JavaScript，目的是为了利用 Java 这个因特网时髦词汇。Netscape 的赌注最终得到回报，JavaScript 从此变成了因特网的必备组件。

三足鼎立

因为 JavaScript 1.0 如此成功，Netscape 在 Netscape Navigator 3.0 中发布了 1.1 版。恰巧那个时候，微软决定进军浏览器，发布了 IE 3.0 并搭载了一个 JavaScript 的克隆版，叫做 JScript（这样命名是为了避免与 Netscape 潜在的许可纠纷）。微软步入 Web 浏览器领域的这重要一步虽然令其声名狼藉，但也成为 JavaScript 语言发展过程中的重要一步。

在微软进入后，有 3 种不同的 JavaScript 版本同时存在：Netscape Navigator 3.0 中的 JavaScript、IE 中的 JScript 以及 CEnvi 中的 ScriptEase。与 C 和其他编程语言不同的是，JavaScript 并没有一个标准来统一其语法或特性，而这 3 种不同的版本恰恰突出了这个问题。随着业界担心的增加，这个语言的标准化显然已经势在必行。

标准化

1997 年，JavaScript 1.1 作为一个草案提交给欧洲计算机制造商协会（ECMA）。第 39 技术委员会（TC39）被委派来“标准化一个通用、跨平台、中立于厂商的脚本语言的语法和语义”(http://www.ecma-international.org/memento/TC39.htm)。由来自 Netscape、Sun、微软、Borland 和其他一些对脚本编程感兴趣的公司的程序员组成的 TC39 锤炼出了 ECMA-262，该标准定义了名为 ECMAScript 的全新脚本语言。

在接下来的几年里，国际标准化组织及国际电工委员会（ISO/IEC）也采纳 ECMAScript 作为标准（ISO/IEC-16262）。从此，Web 浏览器就开始努力（虽然有着不同的程度的成功和失败）将 ECMAScript 作为 JavaScript 实现的基础。

JavaScript 实现

JavaScript 的核心 **ECMAScript** 描述了该语言的语法和基本对象；

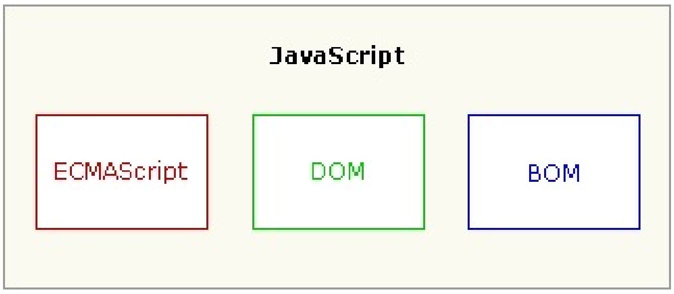
DOM 描述了处理网页内容的方法和接口；

BOM 描述了与浏览器进行交互的方法和接口。

ECMAScript、DOM 和 BOM

尽管 ECMAScript 是一个重要的标准，但它并不是 JavaScript 唯一的部分，当然，也不是唯一被标准化的部分。实际上，一个完整的 JavaScript 实现是由以下 3 个不同部分组成的：

- 核心（ECMAScript）
- 文档对象模型（DOM）
- 浏览器对象模型（BOM）



ECMAScript

ECMAScript 并不与任何具体浏览器相绑定，实际上，它也没有提到用于任何用户输入输出的方法（这点与 C 这类语言不同，它需要依赖外部的库来完成这类任务）。那么什么才是 ECMAScript 呢？ECMA-262 标准（第 2 段）的描述如下：

> “ECMAScript 可以为不同种类的宿主环境提供核心的脚本编程能力，因此核心的脚本语言是与任何特定的宿主环境分开进行规定的... ...”

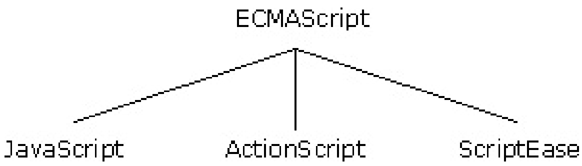
Web 浏览器对于 ECMAScript 来说是一个宿主环境，但它并不是唯一的宿主环境。事实上，还有不计其数的其他各种环境（例如 Nombas 的 ScriptEase，以及 Macromedia 同时用在 Flash 和 Director MX 中的 ActionScript）可以容纳 ECMAScript 实现。那么 ECMAScript 在浏览器之外规定了些什么呢？

简单地说，ECMAScript 描述了以下内容：

- 语法
- 类型
- 语句
- 关键字

- 保留字
- 运算符
- 对象

ECMAScript 仅仅是一个描述，定义了脚本语言的所有属性、方法和对象。其他语言可以实现 ECMAScript 来作为功能的基准，JavaScript 就是这样：



每个浏览器都有它自己的 ECMAScript 接口的实现，然后这个实现又被扩展，包含了 DOM 和 BOM（在以下几节中再探讨）。当然还有其他实现并扩展了 ECMAScript 的语言，例如 Windows 脚本宿主（Windows Scripting Host, WSH）、Macromedia 在 Flash 和 Director MX 中的 ActionScript，以及 Nombas ScriptEase。

1. ECMAScript 的版本

ECMAScript 分成几个不同的版本，它是在一个叫做 ECMA-262 的标准中定义的。和其他标准一样，ECMA-262 会被编辑和更新。当有了主要更新时，就会发布一个标准的新版。最新 ECMA-262 的版本是 5.1，于 2011 年 6 月发布。

ECMA-262 的第一版在本质上与 Netscape 的 JavaScript 1.1 是一样，只是把所有与浏览器相关的代码删除了，此外还有一些小的调整。首先，ECMA-262 要求对 Unicode 标准的支持（以便支持多语言）。第二，它要求对象是平台无关的（Netscape 的 JavaScript 1.1 事实上有不同的对象实现，例如 Date 对象，是依赖于平台）。这是 JavaScript 1.1 和 1.2 为什么不符合 ECMA-262 规范第一版的主要原因。

ECMA-262 的第二版大部分更新本质上是编辑性的。这次标准的更新是为了与 ISO/IEC-16262 的严格一致，也并没有特别添加、更改和删除内容。ECMAScript 一般不会遵守第二版。

ECMA-262 第三版是该标准第一次真正的更新。它提供了对字符串处理、错误定义和数值输出的更新。同时，它还增加了正则表达式、新的控制语句、try...catch 异常处理的支持，以及一些为使标准国际化而做的小改动。一般来说，它标志着 ECMAScript 成为一种真正的编程语言。

2. 何谓 ECMAScript 符合性

在 ECMA-262 中，ECMAScript 符合性（conformance）有明确的定义。一个脚本语言必须满足以下四项基本原则：

- 符合的实现必须按照 ECMA-262 中所描述的支持所有的“类型、值、对象、属性、函数和程序语言及语义”（ECMA-262，第一页）
- 符合的实现必须支持 Unicode 字符标准（UCS）
- 符合的实现可以增加没有在 ECMA-262 中指定的“额外类型、值、对象、属性

和函数”。ECMA-262 将这些增加描述为规范中未给定的新对象或对象的新属性
- 符合的实现可以支持没有在 ECMA-262 中定义的“程序和正则表达式语法”（意思是可以替换或者扩展内建的正则表达式支持）

所有 ECMAScript 实现必须符合以上标准。

3. Web 浏览器中的 ECMAScript 支持

含有 JavaScript 1.1 的 Netscape Navigator 3.0 在 1996 年发布。然后，JavaScript 1.1 规范被作为一个新标准的草案被提交给 EMCA。有了 JavaScript 轰动性的流行，Netscape 十分高兴地开始开发 1.2 版。但有一个问题，ECMA 并未接受 Netscape 的草案。在 Netscape Navigator 3.0 发布后不久，微软就发布了 IE 3.0。该版本的 IE 含有 JScript 1.0（微软自己的 JavaScript 实现的名称），原本计划可以与 JavaScript 1.1 相提并论。然后，由于文档不全以及一些不当的重复特性，JScript 1.0 远远没有达到 JavaScript 1.1 的水平。

在 ECMA-262 第一版定稿之前，发布含有 JavaScript 1.2 的 Netscape Navigator 4.0 是在 1997 年，在那年晚些时候，ECMA-262 标准被接受并标准化。因此，JavaScript 1.2 并不和 ECMAScript 的第一版兼容，虽然 ECMAScript 应该基于 JavaScript 1.1。

JScript 的下一步是 IE 4.0 中加入的 JScript 3.0（2.0 版是随 IIS 3.0 一起发布的，但并未包含在浏览器中）。微软大力宣传 JScript 3.0 是世界上第一个真正符合 ECMA 标准的脚本语言。而那时，ECMA-262 还没有最终定稿，所以 JScript 3.0 也遭受了和 JavaScript 1.2 同样的命运 - 它还是没能符合最终的 ECMAScript 标准。

Netscape 选择在 Netscape Navigator 4.06 中升级它的 JavaScript 实现。JavaScript 1.3 使 Netscape 终于完全符合了 ECMAScript 第一版。Netscape 加入了对 Unicode 标准的支持，并让所有的对象保留了在 JavaScript 1.2 中引入的新特性的同时实现了平台独立。

当 Netscape 将它的源代码作为 Mozilla 项目公布于众时，本来计划 JavaScript 1.4 将会嵌入到 Netscape Navigator 5.0 中。然而，一个冒进的决定 - 要完全从头重新设计 Netscape 的代码，破坏了这个工作。JavaScript 1.4 仅仅作为一个 Netscape Enterprise Server 的服务器端脚本语言发布，以后也没有被放入浏览器中。

如今，所有主流的 Web 浏览器都遵守 ECMA-262 第三版。

下面的表格列出了大部分流行的 Web 浏览器中的 ECMAScript 支持：

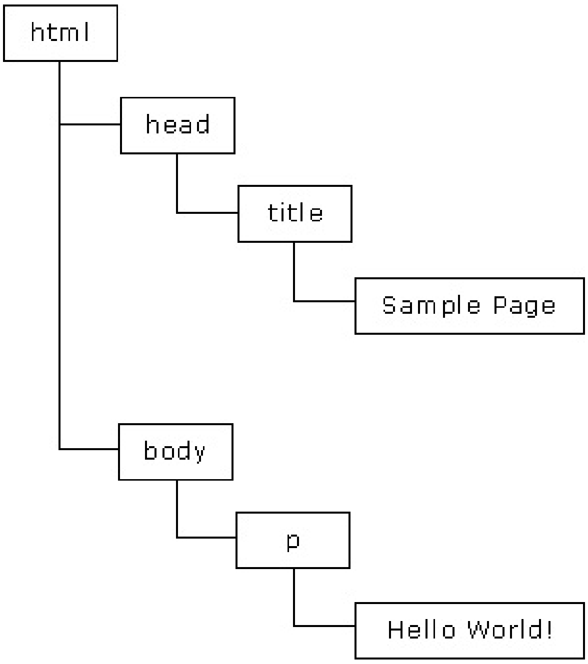
浏览器	DOM 兼容性
Netscape Navigator 2.0	-
Netscape Navigator 3.0	-
Netscape Navigator 4.0 - 4.05	-
Netscape Navigator 4.06 - 4.79	Edition 1
Netscape 6.0+ (Mozilla 0.6.0+)	Edition 3
Internet Explorer 3.0	-
Internet Explorer 4.0	-
Internet Explorer 5.0	Edition 1
Internet Explorer 5.5+	Edition 3
Opera 6.0 - 7.1	Edition 2
Opera 7.2+	Edition 3
Safari 1.0+/Konqueror ~ 2.0+	Edition 3

DOM

DOM（文档对象模型）是 HTML 和 XML 的应用程序接口（API）。DOM 将把整个页面规划成由节点层级构成的文档。HTML 或 XML 页面的每个部分都是一个节点的衍生物。请考虑下面的 HTML 页面：

```
<html>
  <head>
    <title>Sample Page</title>
  </head>
  <body>
    <p>hello world!</p>
  </body>
</html>
```

这段代码可以用 DOM 绘制成一个节点层次图：



DOM 通过创建树来表示文档，从而使开发者对文档的内容和结构具有空前的控制力。用 DOM API 可以轻松地删除、添加和替换节点。

1. 为什么 DOM 必不可少

自从 IE 4.0 和 Netscape Navigator 4.0 开始支持不同形态的动态 HTML（DHTML），开发者首次能够在不重载网页的情况下修改它的外观和内容。这是 Web 技术的一大飞跃，不过也带来了巨大的问题。Netscape 和微软各自开发自己的 DHTML，从而结束了 Web 开发者只编写一个 HTML 页面就可以在所有浏览器中访问的时期。

业界决定必须要做点什么以保持 Web 的跨平台特性，他们担心如果放任 Netscape 和微软公司这样做，Web 必将分化为两个独立的部分，每一部分只适用于特定的浏览器。因此，负责指定 Web 通信标准的团体 W3C（World Wide Web Consortium）就开始制定 DOM。

2. DOM 的各个 level

DOM Level 1 是 W3C 于 1998 年 10 月提出的。它由两个模块组成，即 DOM Core 和 DOM HTML。前者提供了基于 XML 的文档的结构图，以便访问和操作文档的任意部分；后者添加了一些 HTML 专用的对象和方法，从而扩展了 DOM Core。

注意，DOM 不是 JavaScript 专有的，事实上许多其他语言都实现了它。不过，Web 浏览器中的 DOM 已经用 ECMAScript 实现了，现在是 JavaScript 语言的一个很大组成部分。

DOM Level 1 只是一个目标，即规划文档的结构，DOM Level 2 的目标就广泛多了。对原始 DOM 的扩展添加了对鼠标和用户界面事件（DHTML 对此有丰富的支持）、范围、遍历（重复执行 DOM 文档的方法）的支持，并通过对象接口添加了对 CSS（层叠样式表）的支持。由 Level 1 引入的原始 DOM Core 也加入了对 XML 命名空间的支持。

DOM Level 2 引入了几种 DOM 新模块，用于处理新的接口类型：

- DOM 视图 - 描述跟踪文档的各种视图（即 CSS 样式化之前和 CSS 样式化之后的文档）
- DOM 事件 - 描述事件的接口
- DOM 样式 - 描述处理基于 CSS 样式的接口
- DOM 遍历和范围 - 描述遍历和操作文档树的接口

DOM Level 3 引入了以统一的方式载入和保持文档的方法（包含在新模块 DOM Load and Save）以及验证文档（DOM Validation）的方法，从而进一步扩展了 DOM。在 Level 3 中，DOM Core 被扩展为支持所有的 XML 1.0 特性，包括 XML Infoset、XPath 和 XML Base。

在学习 DOM 时，可能会遇到有人引用 DOM Level 0。注意，根本没有 DOM Level 0 这个标准，它只是 DOM 的一个历史参考点（DOM Level 0 指的是 IE 4.0 和 Netscape Navigator 4.0 中支持的原始 DHTML）。

3. 其他 DOM

除了 DOM Core 和 DOM HTML 外，还有其他几种语言发布了自己的 DOM 标准。这些语言都是基于 XML 的，每种 DOM 都给对应语言添加了特有的方法和接口：

- 可缩放矢量语言（SVG）1.0
- 数字标记语言（MathML）1.0
- 同步多媒体集成语言（SMIL）

注释：如果希望学习更多相关内容，请访问 w3school 的 SMIL 教程 和 SVG 教程。

此外，其他语言也开发了自己的 DOM 实现，如 Mozilla 的 XML 用户界面语言（XUL）。不过，只有上面列出的几种语言是 W3C 的推荐标准。

4. Web 浏览器中的 DOM 支持

DOM 在被 Web 浏览器开始实现之前就已经是一种标准了。IE 首次尝试 DOM 是在 5.0 版本中，不过其实直到 5.5 版本之后才具有真正的 DOM 支持，IE 5.5 实现了 DOM Level 1。从那时起，IE 就没有引入新的 DOM 功能。

Netscape 直到 Netscape 6（Mozilla 0.6.0）才引入 DOM 支持。目前，Mozilla 具有最好的 DOM 支持，实现了完整的 Level 1、几乎所有 Level 2 以及一部分 Level 3。（Mozilla 开发小组的目标是构造一个与标准 100% 兼容的浏览器，他们的工作得到了回报。）

Opera 直到 7.0 版本才加入 DOM 支持，还有 Safari 也实现了大部分 DOM Level 1。它们几乎都与 IE 5.5 处于同一水平，有些情况下，甚至超过了 IE 5.5。不过，就对 DOM 的支持而论，所有浏览器都远远落后于 Mozilla。下表列出了常用浏览器对 DOM 的支持。

浏览器	DOM 兼容性
Netscape Navigator 1.0 - 4.x	-
Netscape 6.0+ (Mozilla 0.6.0+)	Level 1、Level 2、Level 3（部分）
IE 2.0 - 4.x	-
IE 5.0	Level 1（最小）
IE 5.5+	Level 1（几乎全部）
Opera 1.0 - 6.0	-
Opera 7.0+	Level 1（几乎全部）、Level 2 （部分）
Safari 1.0+/Konqueror ~ 2.0+	Level 1

注释：如果希望进一步地学习 DOM 的知识，请访问 w3school 的 HTML DOM 教程 和 XML DOM 教程。

BOM

IE 3.0 和 Netscape Navigator 3.0 提供了一种特性 - BOM（浏览器对象模型），可以对浏览器窗口进行访问和操作。使用 BOM，开发者可以移动窗口、改变状态栏中的文本以及执行其他与页面内容不直接相关的动作。使 BOM 独树一帜且又常常令人怀疑的地方在于，它只是 JavaScript 的一个部分，没有任何相关的标准。

BOM 主要处理浏览器窗口和框架，不过通常浏览器特定的 JavaScript 扩展都被看做 BOM 的一部分。这些扩展包括：

- 弹出新的浏览器窗口
- 移动、关闭浏览器窗口以及调整窗口大小
- 提供 Web 浏览器详细信息的定位对象
- 提供用户屏幕分辨率详细信息的屏幕对象
- 对 cookie 的支持
- IE 扩展了 BOM，加入了 ActiveXObject 类，可以通过 JavaScript 实例化 ActiveX 对象

由于没有相关的 BOM 标准，每种浏览器都有自己的 BOM 实现。有一些事实上的标准，如具有一个窗口对象和一个导航对象，不过每种浏览器可以为这些对象或其他对象定义自己的属性和方法。

参阅：

- Window 对象
- Navigator 对象
- Screen 对象
- History 对象
- Location 对象

ECMAScript 基础

ECMAScript 语法

熟悉 **Java**、**C** 和 **Perl** 这些语言的开发者会发现 **ECMAScript** 的语法很容易掌握，因为它借用了这些语言的语法。

Java 和 **ECMAScript** 有一些关键的语法特性相同，也有一些完全不同。

区分大小写

与 Java 一样，变量、函数名、运算符以及其他一切东西都是区分大小写的。

比如：

变量 test 与变量 TEST 是不同的。

变量是弱类型的

与 Java 和 C 不同，ECMAScript 中的变量无特定的类型，定义变量时只用 var 运算符，可以将它初始化为任意值。

因此，可以随时改变变量所存数据的类型（尽量避免这样做）。

例子

```
var color = "red";
var num = 25;
var visible = true;
```

每行结尾的分号可有可无

Java、C 和 Perl 都要求每行代码以分号（;）结束才符合语法。

ECMAScript 则允许开发者自行决定是否以分号结束一行代码。如果没有分号，ECMAScript 就把折行代码的结尾看做该语句的结尾（与 Visual Basic 和 VBScript 相似），前提是这样没有破坏代码的语义。

最好的代码编写习惯是总加入分号，因为没有分号，有些浏览器就不能正确运行，不过根据 ECMAScript 标准，下面两行代码都是正确的：

```
var test1 = "red"
var test2 = "blue";
```

注释与 Java、C 和 PHP 语言的注释相同

ECMAScript 借用了这些语言的注释语法。

有两种类型的注释：

- 单行注释以双斜杠开头（//）
- 多行注释以单斜杠和星号开头（/），以星号和单斜杠结尾（/）

```
//this is a single-line comment

/*this is a multi-
line comment*/
```

括号表示代码块

从 Java 中借鉴的另一个概念是代码块。

代码块表示一系列应该按顺序执行的语句，这些语句被封装在左括号（{）和右括号（}）之间。

例如：

```
if (test1 == "red") {
    test1 = "blue";
    alert(test1);
}
```

ECMAScript 变量

请使用 **var** 运算符声明变量。

变量名需要遵守一些简单的规则。

声明变量

在上一节中我们讲解过，ECMAScript 中的变量是用 var 运算符（variable 的缩写）加变量名定义的。例如：

```
var test = "hi";
```

在这个例子中，声明了变量 test，并把它的值初始化为 "hi"（字符串）。由于 ECMAScript 是弱类型的，所以解释程序会为 test 自动创建一个字符串值，无需明确的类型声明。

还可以用一个 var 语句定义两个或多个变量：

```
var test1 = "hi", test2 = "hello";
```

前面的代码定义了变量 test1，初始值为 "hi"，还定义了变量 test2，初始值为 "hello"。

不过用同一个 *var* 语句定义的变量不必具有相同的类型，如下所示：

```
var test = "hi", age = 25;
```

这个例子除了（再次）定义 test 外，还定义了 age，并把它初始化为 25。即使 test 和 age 属于两种不同的数据类型，在 ECMAScript 中这样定义也是完全合法的。

与 Java 不同，ECMAScript 中的变量并不一定要初始化（它们是在幕后初始化的，将在后面讨论这一点）。因此，下面这一行代码也是有效的：

```
var test;
```

此外，与 Java 不同的还有变量可以存放不同类型的值。这是弱类型变量的优势。例如，可以把变量初始化为字符串类型的值，之后把它设置为数字值，如下所示：

```
var test = "hi";
alert(test);
test = 55;
alert(test);
```

这段代码将毫无问题地输出字符串值和数字值。但是，如前所述，使用变量时，好的编码习惯是始终存放相同类型的值。

命名变量

变量名需要遵守两条简单的规则：

- 第一个字符必须是字母、下划线（_）或美元符号（$）
- 余下的字符可以是下划线、美元符号或任何字母或数字字符

下面的变量都是合法的：

```
var test;
var $test;
var $1;
var _$te$t2;
```

著名的变量命名规则

只是因为变量名的语法正确，并不意味着就该使用它们。变量还应遵守以下某条著名的命名规则：

Camel 标记法

首字母是小写的，接下来的字母都以大写字符开头。例如：

```
var myTestValue = 0, mySecondValue = "hi";
```

Pascal 标记法

首字母是大写的，接下来的字母都以大写字符开头。例如：

```
var MyTestValue = 0, MySecondValue = "hi";
```

匈牙利类型标记法

在以 Pascal 标记法命名的变量前附加一个小写字母（或小写字母序列），说明该变量的类型。例如，i 表示整数，s 表示字符串，如下所示“

```
var iMyTestValue = 0, sMySecondValue = "hi";
```

本教程采用了这些前缀，以使示例代码更易阅读：

类型	前缀	示例
数组	a	aValues
布尔型	b	bFound
浮点型（数字）	f	fValue
函数	fn	fnMethod
整型（数字）	i	iValue
对象	o	oType
正则表达式	re	rePattern
字符串	s	sValue
变型（可以是任何类型）	v	vValue

变量声明不是必须的

ECMAScript 另一个有趣的方面（也是与大多数程序设计语言的主要区别），是在使用变量之前不必声明。例如：

```
var sTest = "hello ";
sTest2 = sTest + "world";
alert(sTest2);
```

在上面的代码中，首先，sTest 被声明为字符串类型的值 "hello"。接下来的一行，用变量 sTest2 把 sTest 与字符串 "world" 连在一起。变量 sTest2 并没有用 var 运算符定义，这里只是插入了它，就像已经声明过它一样。

ECMAScript 的解释程序遇到未声明过的标识符时，用该变量名创建一个全局变量，并将其初始化为指定的值。

这是该语言的便利之处，不过如果不能紧密跟踪变量，这样做也很危险。最好的习惯是像使用其他程序设计语言一样，总是声明所有变量。

ECMAScript 关键字

本节提供完整的 **ECMAScript** 关键字列表。

ECMAScript 关键字

ECMA-262 定义了 ECMAScript 支持的一套关键字（*keyword*）。

这些关键字标识了 ECMAScript 语句的开头和/或结尾。根据规定，关键字是保留的，不能用作变量名或函数名。

下面是 ECMAScript 关键字的完整列表：

```
break
case
catch
continue
default
delete
do
else
finally
for
function
if
in
instanceof
new
return
switch
this
throw
try
typeof
var
void
while
with
```

注意：如果把关键字用作变量名或函数名，可能得到诸如 "Identifier Expected"（应该有标识符、期望标识符）这样的错误消息。

ECMAScript 保留字

本节提供完整的 **ECMAScript** 保留字列表。

ECMAScript 保留字

ECMA-262 定义了 ECMAScript 支持的一套保留字（*reserved word*）。

保留字在某种意思上是为将来的关键字而保留的单词。因此保留字不能被用作变量名或函数名。

ECMA-262 第三版中保留字的完整列表如下：

```
abstract
boolean
byte
char
class
const
debugger
double
enum
export
extends
final
float
goto
implements
import
int
interface
long
native
package
private
protected
public
short
static
super
synchronized
throws
transient
volatile
```

注意：如果将保留字用作变量名或函数名，那么除非将来的浏览器实现了该保留字，否则很可能收不到任何错误消息。当浏览器将其实现后，该单词将被看做关键字，如此将出现关键字错误。

ECMAScript 原始值和引用值

在 **ECMAScript** 中，变量可以存在两种类型的值，即原始值和引用值。

原始值和引用值

在 ECMAScript 中，变量可以存在两种类型的值，即原始值和引用值。

原始值

存储在栈（stack）中的简单数据段，也就是说，它们的值直接存储在变量访问的位置。

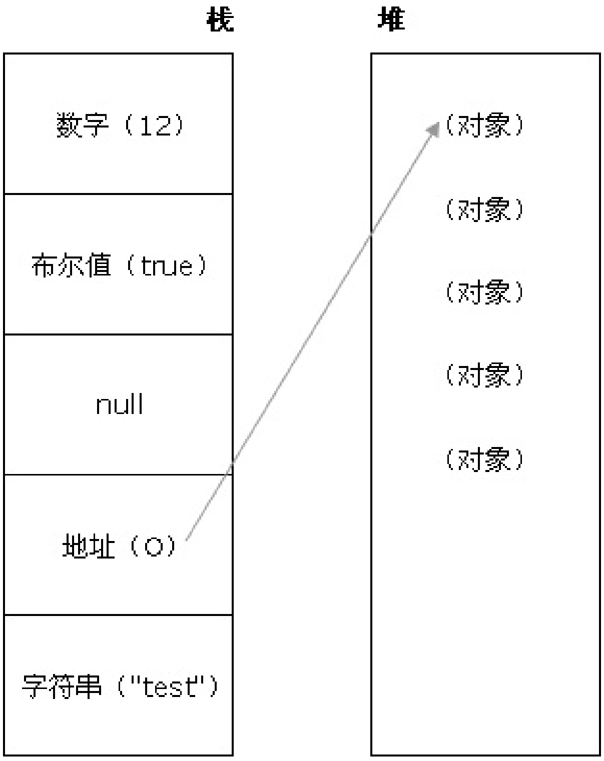
引用值

存储在堆（heap）中的对象，也就是说，存储在变量处的值是一个指针（point），指向存储对象的内存处。

为变量赋值时，ECMAScript 的解释程序必须判断该值是原始类型，还是引用类型。要实现这一点，解释程序则需尝试判断该值是否为 *ECMAScript* 的原始类型之一，即 Undefined、Null、Boolean、Number 和 String 型。由于这些原始类型占据的空间是固定的，所以可将他们存储在较小的内存区域 - 栈中。这样存储便于迅速查寻变量的值。

在许多语言中，字符串都被看作引用类型，而非原始类型，因为字符串的长度是可变的。**ECMAScript** 打破了这一传统。

如果一个值是引用类型的，那么它的存储空间将从堆中分配。由于引用值的大小会改变，所以不能把它放在栈中，否则会降低变量查寻的速度。相反，放在变量的栈空间中的值是该对象存储在堆中的地址。地址的大小是固定的，所以把它存储在栈中对变量性能无任何负面影响。如下图所示：



原始类型

如前所述，ECMAScript 有 5 种原始类型（primitive type），即 Undefined、Null、Boolean、Number 和 String。ECMA-262 把术语类型（*type*）定义为值的一个集合，每种原始类型定义了它包含的值的范围及其字面量表示形式。

ECMAScript 提供了 typeof 运算符来判断一个值是否在某种类型的范围内。可以用这种运算符判断一个值是否表示一种原始类型：如果它是原始类型，还可以判断它表示哪种原始类型。

在稍后的章节，我们将为您深入讲解 ECMAScript 的原始类型和引用类型。

ECMAScript 原始类型

ECMAScript 有 5 种原始类型（primitive type），即 Undefined、Null、Boolean、Number 和 String。

typeof 运算符

typeof 运算符有一个参数，即要检查的变量或值。例如：

```
var sTemp = "test string";
alert (typeof sTemp);    //输出 "string"
alert (typeof 86);    //输出 "number"
```

对变量或值调用 typeof 运算符将返回下列值之一：

- undefined - 如果变量是 Undefined 类型的
- boolean - 如果变量是 Boolean 类型的
- number - 如果变量是 Number 类型的
- string - 如果变量是 String 类型的
- object - 如果变量是一种引用类型或 Null 类型的

注释：您也许会问，为什么 typeof 运算符对于 null 值会返回 "Object"。这实际上是 JavaScript 最初实现中的一个错误，然后被 ECMAScript 沿用了。现在，null 被认为是对象的占位符，从而解释了这一矛盾，但从技术上来说，它仍然是原始值。

Undefined 类型

如前所述，Undefined 类型只有一个值，即 undefined。当声明的变量未初始化时，该变量的默认值是 undefined。

```
var oTemp;
```

前面一行代码声明变量 oTemp，没有初始值。该变量将被赋予值 undefined，即 undefined 类型的字面量。可以用下面的代码段测试该变量的值是否等于 undefined：

```
var oTemp;
alert(oTemp == undefined);
```

这段代码将显示 "true"，说明这两个值确实相等。还可以用 typeof 运算符显示该变量的值是 undefined：

```
var oTemp;
alert(typeof oTemp); //输出 "undefined"
```

提示：值 undefined 并不同于未定义的值。但是，typeof 运算符并不真正区分这两种值。考虑下面的代码：

```
var oTemp;

alert(typeof oTemp);  //输出 "undefined"
alert(typeof oTemp2);  //输出 "undefined"
```

前面的代码对两个变量输出的都是 "undefined"，即使只有变量 oTemp2 从未被声明过。如果对 oTemp2 使用除 typeof 之外的其他运算符的话，会引起错误，因为其他运算符只能用于已声明的变量上。

例如，下面的代码将引发错误：

```
var oTemp;
alert(oTemp2 == undefined);
```

当函数无明确返回值时，返回的也是值 "undefined"，如下所示：

```
function testFunc() {
}

alert(testFunc() == undefined);  //输出 "true"
```

Null 类型

另一种只有一个值的类型是 Null，它只有一个专用值 null，即它的字面量。值 undefined 实际上是从值 null 派生来的，因此 ECMAScript 把它们定义为相等的。

```
alert(null == undefined);  //输出 "true"
```

尽管这两个值相等，但它们的含义不同。undefined 是声明了变量但未对其初始化时赋予该变量的值，null 则用于表示尚未存在的对象（在讨论 typeof 运算符时，简单地介绍过这一点）。如果函数或方法要返回的是对象，那么找不到该对象时，返回的通常是 null。

Boolean 类型

Boolean 类型是 ECMAScript 中最常用的类型之一。它有两个值 true 和 false （即两个 Boolean 字面量）。

即使 false 不等于 0，0 也可以在必要时被转换成 false，这样在 Boolean 语句中使用两者都是安全的。

```
var bFound = true;
var bLost = false;
```

Number 类型

ECMA-262 中定义的最特殊的类型是 Number 类型。这种类型既可以表示 32 位的整数，还可以表示 64 位的浮点数。

直接输入的（而不是从另一个变量访问的）任何数字都被看做 Number 类型的字面量。例如，下面的代码声明了存放整数值的变量，它的值由字面量 86 定义：

```
var iNum = 86;
```

八进制数和十六进制数

整数也可以被表示为八进制（以 8 为底）或十六进制（以 16 为底）的字面量。八进制字面量的首数字必须是 0，其后的数字可以是任何八进制数字（0-7），如下面的代码所示：

```
var iNum = 070;  //070 等于十进制的 56
```

要创建十六进制的字面量，首位数字必须为 0，后面接字母 x，然后是任意的十六进制数字（0 到 9 和 A 到 F）。这些字母可以是大写的，也可以是小写的。例如：

```
var iNum = 0x1f;  //0x1f 等于十进制的 31
var iNum = 0xAB;  //0xAB 等于十进制的 171
```

提示：尽管所有整数都可以表示为八进制或十六进制的字面量，但所有数学运算返回的都是十进制结果。

浮点数

要定义浮点值，必须包括小数点和小数点后的一位数字（例如，用 1.0 而不是 1）。这被看作浮点数字面量。例如：

```
var fNum = 5.0;
```

对于浮点字面量的有趣之处在于，用它进行计算前，真正存储的是字符串。

科学计数法

对于非常大或非常小的数，可以用科学计数法表示浮点数，可以把一个数表示为数字（包括十进制数字）加 e（或 E），后面加乘以 10 的倍数。例如：

```
var fNum = 5.618e7
```

该符号表示的是数 56180000。把科学计数法转化成计算式就可以得到该值：5.618 x 10⁷。

也可以用科学计数法表示非常小的数，例如 0.00000000000000008 可以表示为 8-e¹⁷（这里，10 被升到 -17 次冥，意味着需要被 10 除 17 次）。ECMAScript 默认把具有 6 个或 6 个以上前导 0 的浮点数转换成科学计数法。

提示：也可用 64 位 IEEE 754 形式存储浮点值，这意味着十进制值最多可以有 17 个十进制位。17 位之后的值将被裁去，从而造成一些小的数学误差。

特殊的 **Number** 值

几个特殊值也被定义为 Number 类型。前两个是 Number.MAX_VALUE 和 Number.MIN_VALUE，它们定义了 Number 值集合的外边界。所有 ECMAScript 数都必须在这两个值之间。不过计算生成的数值结果可以不落在这两个值之间。

当计算生成的数大于 Number.MAX_VALUE 时，它将被赋予值 Number.POSITIVE_INFINITY，意味着不再有数字值。同样，生成的数值小于 Number.MIN_VALUE 的计算也会被赋予值 Number.NEGATIVE_INFINITY，也意味着不再有数字值。如果计算返回的是无穷大值，那么生成的结果不能再用于其他计算。

事实上，有专门的值表示无穷大，（如你猜到的）即 Infinity。Number.POSITIVE_INFINITY 的值为 Infinity。Number.NEGATIVE_INFINITY 的值为 -Infinity。

由于无穷大数可以是正数也可以是负数，所以可用一个方法判断一个数是否是有穷的（而不是单独测试每个无穷数）。可以对任何数调用 isFinite() 方法，以确保该数不是无穷大。例如：

```
var iResult = iNum * some_really_large_number;

`if (isFinite(iResult))` {
    alert("finite");
}

else {
    alert("infinite");
}
```

最后一个特殊值是 NaN，表示非数（Not a Number）。NaN 是个奇怪的特殊值。一般说来，这种情况发生在类型（String、Boolean 等）转换失败时。例如，要把单词 blue 转换成数值就会失败，因为没有与之等价的数值。与无穷大一样，NaN 也不能用于算术计算。NaN 的另一个奇特之处在于，它与自身不相等，这意味着下面的代码将返回 false：

```
alert(NaN == NaN);  //输出 "false"
```

出于这个原因，不推荐使用 NaN 值本身。函数 isNaN() 会做得相当好：

```
alert(isNaN("blue"));  //输出 "true"
alert(isNaN("666"));  //输出 "false"
```

String 类型

String 类型的独特之处在于，它是唯一没有固定大小的原始类型。可以用字符串存储 0 或更多的 Unicode 字符，有 16 位整数表示（Unicode 是一种国际字符集，本教程后面将讨论它）。

字符串中每个字符都有特定的位置，首字符从位置 0 开始，第二个字符在位置 1，依此类推。这意味着字符串中的最后一个字符的位置一定是字符串的长度减 1：



字符串字面量是由双引号（"）或单引号（'）声明的。而 Java 则是用双引号声明字符串，用单引号声明字符。但是由于 ECMAScript 没有字符类型，所以可使用这两种表示法中的任何一种。例如，下面的两行代码都有效：

```
var sColor1 = "red";
var sColor2 = 'red';
```

String 类型还包括几种字符字面量，Java、C 和 Perl 的开发者应该对此非常熟悉。

下面列出了 ECMAScript 的字符字面量：

字面量	含义
\n	换行
\t	制表符
\b	空格
\r	回车
\f	换页符
\	反斜杠
\'	单引号
\"	双引号
\0*nnn*	八进制代码 *nnn* 表示的字符（*n* 是 0 到 7 中的一个八进制数字）
\x*nn*	十六进制代码 *nn* 表示的字符（*n* 是 0 到 F 中的一个十六进制数字）
\u*nnnn*	十六进制代码 *nnnn* 表示的 Unicode 字符（*n* 是 0 到 F 中的一个十六进制数字）

ECMAScript 类型转换

所有程序设计语言最重要的特征之一是具有进行类型转换的能力。

ECMAScript 给开发者提供了大量简单的类型转换方法。

大部分类型具有进行简单转换的方法，还有几个全局方法可以用于更复杂的转换。无论哪种情况，在 **ECMAScript** 中，类型转换都是简短的一步操作。

转换成字符串

ECMAScript 的 Boolean 值、数字和字符串的原始值的有趣之处在于它们是伪对象，这意味着它们实际上具有属性和方法。

例如，要获得字符串的长度，可以采用下面的代码：

```
var sColor = "red";
alert(sColor.length);    //输出 "3"
```

尽管 "red" 是原始类型的字符串，它仍然具有属性 length，用于存放字符串的大小。

总而言之，3 种主要的原始类型 Boolean 值、数字和字符串都有 toString() 方法，可以把它们的值转换成字符串。

提示：您也许会问，“字符串还有 toString() 方法吗，这不是多余吗？”是的，的确如此，不过 ECMAScript 定义所有对象都有 toString() 方法，无论它是伪对象，还是真对象。因为 String 类型属于伪对象，所以它一定有 toString() 方法。

Boolean 类型的 toString() 方法只是输出 "true" 或 "false"，结果由变量的值决定：

```
var bFound = false;
alert(bFound.toString());    //输出 "false"
```

Number 类型的 toString() 方法比较特殊，它有两种模式，即默认模式和基模式。采用默认模式，toString() 方法只是用相应的字符串输出数字值（无论是整数、浮点数还是科学计数法），如下所示：

```
var iNum1 = 10;
var iNum2 = 10.0;
alert(iNum1.toString());    //输出 "10"
alert(iNum2.toString());    //输出 "10"
```

注释：在默认模式中，无论最初采用什么表示法声明数字，Number 类型的 toString() 方法返回的都是数字的十进制表示。因此，以八进制或十六进制字面量形式声明的数字输出的都是十进制形式的。

采用 Number 类型的 toString() 方法的基模式，可以用不同的基输出数字，例如二进制的基是 2，八进制的基是 8，十六进制的基是 16。

基只是要转换成的基数的另一种加法而已，它是 toString() 方法的参数：

```
var iNum = 10;
alert(iNum.toString(2));    //输出 "1010"
alert(iNum.toString(8));    //输出 "12"
alert(iNum.toString(16));    //输出 "A"
```

在前面的示例中，以 3 种不同的形式输出了数字 10，即二进制形式、八进制形式和十六进制形式。HTML 采用十六进制表示每种颜色，在 HTML 中处理数字时这种功能非常有用。

注释：对数字调用 toString(10) 与调用 toString() 相同，它们返回的都是该数字的十进制形式。

参阅：

请参阅 JavaScript 参考手册提供的有关 toString() 方法的详细信息：

- arrayObject.toString()
- booleanObject.toString()
- dateObject.toString()
- NumberObject.toString()
- stringObject.toString()

转换成数字

ECMAScript 提供了两种把非数字的原始值转换成数字的方法，即 parseInt() 和 parseFloat()。

正如您可能想到的，前者把值转换成整数，后者把值转换成浮点数。只有对 String 类型调用这些方法，它们才能正确运行；对其他类型返回的都是 NaN。

parseInt()

在判断字符串是否是数字值前，parseInt() 和 parseFloat() 都会仔细分析该字符串。

parseInt() 方法首先查看位置 0 处的字符，判断它是否是个有效数字；如果不是，该方法将返回 NaN，不再继续执行其他操作。但如果该字符是有效数字，该方法将查看位置 1 处的字符，进行同样的测试。这一过程将持续到发现非有效数字的字符

为止，此时 parseInt() 将把该字符之前的字符串转换成数字。

例如，如果要把字符串 "12345red" 转换成整数，那么 parseInt() 将返回 12345，因为当它检查到字符 r 时，就会停止检测过程。

字符串中包含的数字字面量会被正确转换为数字，比如 "0xA" 会被正确转换为数字 10。不过，字符串 "22.5" 将被转换成 22，因为对于整数来说，小数点是无效字符。

一些示例如下：

```
var iNum1 = parseInt("12345red");	//返回 12345
var iNum1 = parseInt("0xA");	//返回 10
var iNum1 = parseInt("56.9");	//返回 56
var iNum1 = parseInt("red");	//返回 NaN
```

parseInt() 方法还有基模式，可以把二进制、八进制、十六进制或其他任何进制的字符串转换成整数。基是由 parseInt() 方法的第二个参数指定的，所以要解析十六进制的值，需如下调用 parseInt() 方法：

```
var iNum1 = parseInt("AF", 16);	//返回 175
```

当然，对二进制、八进制甚至十进制（默认模式），都可以这样调用 parseInt() 方法：

```
var iNum1 = parseInt("10", 2);	//返回 2
var iNum2 = parseInt("10", 8);	//返回 8
var iNum3 = parseInt("10", 10);	//返回 10
```

如果十进制数包含前导 0，那么最好采用基数 10，这样才不会意外地得到八进制的值。例如：

```
var iNum1 = parseInt("010");	//返回 8
var iNum2 = parseInt("010", 8);	//返回 8
var iNum3 = parseInt("010", 10);	//返回 10
```

在这段代码中，两行代码都把字符 "010" 解析成一个数字。第一行代码把这个字符串看作八进制的值，解析它的方式与第二行代码（声明基数为 8）相同。最后一行代码声明基数为 10，所以 iNum3 最后等于 10。

参阅

请参阅 JavaScript 参考手册提供的有关 parseInt() 方法的详细信息：parseInt()。

parseFloat()

parseFloat() 方法与 parseInt() 方法的处理方式相似，从位置 0 开始查看每个字符，直到找到第一个非有效的字符为止，然后把该字符之前的字符串转换成整数。

不过，对于这个方法来说，第一个出现的小数点是有效字符。如果有两个小数点，第二个小数点将被看作无效的。parseFloat() 会把这个小数点之前的字符转换成数字。这意味着字符串 "11.22.33" 将被解析成 11.22。

使用 parseFloat() 方法的另一不同之处在于，字符串必须以十进制形式表示浮点数，而不是用八进制或十六进制。该方法会忽略前导 0，所以八进制数 0102 将被解析为 102。对于十六进制数 0xA，该方法将返回 NaN，因为在浮点数中，x 不是有效字符。（注释：经测试，具体的浏览器实现会返回 0，而不是 NaN。）

此外，parseFloat() 方法也没有基模式。

下面是使用 parseFloat() 方法的一些示例：

```
var fNum1 = parseFloat("12345red");    //返回 12345
var fNum2 = parseFloat("0xA");    //返回 NaN
var fNum3 = parseFloat("11.2");    //返回 11.2
var fNum4 = parseFloat("11.22.33");    //返回 11.22
var fNum5 = parseFloat("0102");    //返回 102
var fNum1 = parseFloat("red");    //返回 NaN
```

参阅

请参阅 JavaScript 参考手册提供的有关 parseFloat() 方法的详细信息：parseFloat()。

强制类型转换

您还可以使用强制类型转换（*type casting*）来处理转换值的类型。使用强制类型转换可以访问特定的值，即使它是另一种类型的。

编者注：cast 有“铸造”之意，很贴合“强制转换”的意思。

ECMAScript 中可用的 3 种强制类型转换如下：

- Boolean(value) - 把给定的值转换成 Boolean 型；
- Number(value) - 把给定的值转换成数字（可以是整数或浮点数）；
- String(value) - 把给定的值转换成字符串；

用这三个函数之一转换值，将创建一个新值，存放由原始值直接转换成的值。这会造成意想不到的后果。

Boolean() 函数

当要转换的值是至少有一个字符的字符串、非 0 数字或对象时，Boolean() 函数将返回 true。如果该值是空字符串、数字 0、undefined 或 null，它将返回 false。

可以用下面的代码测试 Boolean 型的强制类型转换：

```
var b1 = Boolean("");		//false - 空字符串
var b2 = Boolean("hello");		//true - 非空字符串
var b1 = Boolean(50);		//true - 非零数字
var b1 = Boolean(null);		//false - null
var b1 = Boolean(0);		//false - 零
var b1 = Boolean(new object());	//true - 对象
```

Number() 函数

Number() 函数的强制类型转换与 parseInt() 和 parseFloat() 方法的处理方式相似，只是它转换的是整个值，而不是部分值。

还记得吗，parseInt() 和 parseFloat() 方法只转换第一个无效字符之前的字符串，因此 "1.2.3" 将分别被转换为 "1" 和 "1.2"。

用 Number() 进行强制类型转换，"1.2.3" 将返回 NaN，因为整个字符串值不能转换成数字。如果字符串值能被完整地转换，Number() 将判断是调用 parseInt() 方法还是 parseFloat() 方法。

下表说明了对不同的值调用 Number() 方法会发生的情况：

用法	结果
Number(false)	0
Number(true)	1
Number(undefined)	NaN
Number(null)	0
Number("1.2")	1.2
Number("12")	12
Number("1.2.3")	NaN
Number(new object())	NaN
Number(50)	50

String() 函数

最后一种强制类型转换方法 String() 是最简单的，因为它可把任何值转换成字符串。

要执行这种强制类型转换，只需要调用作为参数传递进来的值的 toString() 方法，即把 12 转换成 "12"，把 true 转换成 "true"，把 false 转换成 "false"，以此类推。

强制转换成字符串和调用 toString() 方法的唯一不同之处在于，对 null 和 undefined 值强制类型转换可以生成字符串而不引发错误：

```
var s1 = String(null);    //"null"
var oNull = null;
var s2 = oNull.toString();    //会引发错误
```

在处理 ECMAScript 这样的弱类型语言时，强制类型转换非常有用，不过应该确保使用值的正确。

ECMAScript 引用类型

引用类型通常叫做类（**class**）。

本教程会讨论大量的 **ECMAScript** 预定义引用类型。

引用类型

引用类型通常叫做类（class），也就是说，遇到引用值，所处理的就是对象。

本教程会讨论大量的 ECMAScript 预定义引用类型。

从现在起，将重点讨论与已经讨论过的原始类型紧密相关的引用类型。

注意：从传统意义上来说，ECMAScript 并不真正具有类。事实上，除了说明不存在类，在 ECMA-262 中根本没有出现“类”这个词。ECMAScript 定义了“对象定义”，逻辑上等价于其他程序设计语言中的类。

提示：本教程将使用术语“对象”。

对象是由 new 运算符加上要实例化的对象的名字创建的。例如，下面的代码创建 Object 对象的实例：

```
var o = new Object();
```

这种语法与 Java 语言的相似，不过当有不止一个参数时，ECMAScript 要求使用括号。如果没有参数，如以下代码所示，括号可以省略：

```
var o = new Object;
```

注意：尽管括号不是必需的，但是为了避免混乱，最好使用括号。

提示：我们会在对象基础这一章中更深入地探讨对象及其行为。

这一节的重点是具有等价的原始类型的引用类型。

Object 对象

Object 对象自身用处不大，不过在了解其他类之前，还是应该了解它。因为 ECMAScript 中的 Object 对象与 Java 中的 java.lang.Object 相似，ECMAScript 中的所有对象都由这个对象继承而来，Object 对象中的所有属性和方法都会出现在其他对象中，所以理解了 Object 对象，就可以更好地理解其他对象。

Object 对象具有下列属性：

constructor

对创建对象的函数的引用（指针）。对于 Object 对象，该指针指向原始的 Object() 函数。

Prototype

对该对象的对象原型的引用。对于所有的对象，它默认返回 Object 对象的一个实例。

Object 对象还具有几个方法：

hasOwnProperty(property)

判断对象是否有某个特定的属性。必须用字符串指定该属性。（例如，o.hasOwnProperty("name")）

IsPrototypeOf(object)

判断该对象是否为另一个对象的原型。

PropertyIsEnumerable

判断给定的属性是否可以用 for...in 语句进行枚举。

ToString()

返回对象的原始字符串表示。对于 Object 对象，ECMA-262 没有定义这个值，所以不同的 ECMAScript 实现具有不同的值。

ValueOf()

返回最适合该对象的原始值。对于许多对象，该方法返回的值都与 ToString() 的返回值相同。

注释：上面列出的每种属性和方法都会被其他对象覆盖。

Boolean 对象

Boolean 对象是 Boolean 原始类型的引用类型。

要创建 Boolean 对象，只需要传递 Boolean 值作为参数：

```
var oBooleanObject = new Boolean(true);
```

Boolean 对象将覆盖 Object 对象的 ValueOf() 方法，返回原始值，即 true 和 false。ToString() 方法也会被覆盖，返回字符串 "true" 或 "false"。

遗憾的是，在 ECMAScript 中很少使用 Boolean 对象，即使使用，也不易理解。

问题通常出现在 Boolean 表达式中使用 Boolean 对象时。例如：

```
var oFalseObject = new Boolean(false);
var bResult = oFalseObject && true;    //输出 true
```

在这段代码中，用 false 值创建 Boolean 对象。然后用这个值与原始值 true 进行 AND 操作。在 Boolean 运算中，false 和 true 进行 AND 操作的结果是 false。不过，在这行代码中，计算的是 oFalseObject，而不是它的值 false。

正如前面讨论过的，在 Boolean 表达式中，所有对象都会被自动转换为 true，所以 oFalseObject 的值是 true。然后 true 再与 true 进行 AND 操作，结果为 true。

注意：虽然你应该了解 Boolean 对象的可用性，不过最好还是使用 Boolean 原始值，避免发生这一节提到的问题。

参阅

如需更多有关 Boolean 对象的信息，请访问 JavaScript Boolean 对象参考手册。

Number 对象

正如你可能想到的，Number 对象是 Number 原始类型的引用类型。要创建 Number 对象，采用下列代码：

```
var oNumberObject = new Number(68);
```

您应该已认出本章前面小节中讨论特殊值（如 Number.MAX_VALUE）时提到的 Number 对象。所有特殊值都是 Number 对象的静态属性。

要得到数字对象的 Number 原始值，只需要使用 valueOf() 方法：

```
var iNumber = oNumberObject.valueOf();
```

当然，Number 类也有 toString() 方法，在讨论类型转换的小节中已经详细讨论过该方法。

除了从 Object 对象继承的标准方法外，Number 对象还有几个处理数值的专用方法。

toFixed() 方法

toFixed() 方法返回的是具有指定位数小数的数字的字符串表示。例如：

```
var oNumberObject = new Number(68);
alert(oNumberObject.toFixed(2));  //输出 "68.00"
```

在这里，toFixed() 方法的参数是 2，说明应该显示两位小数。该方法返回 "68.00"，空的字符串位由 0 来补充。对于处理货币的应用程序，该方法非常有用。toFixed() 方法能表示具有 0 到 20 位小数的数字，超过这个范围的值会引发错误。

toExponential() 方法

与格式化数字相关的另一个方法是 toExponential()，它返回的是用科学计数法表示的数字的字符串形式。

与 toFixed() 方法相似，toExponential() 方法也有一个参数，指定要输出的小数的位数。例如：

```
var oNumberObject = new Number(68);
alert(oNumberObject.toExponential(1));  //输出 "6.8e+1"
```

这段代码的结果是 "6.8e+1"，前面解释过，它表示 6.8x10¹。问题是，如果不知道要用哪种形式（预定形式或指数形式）表示数字怎么办？可以用 toPrecision() 方法。

toPrecision() 方法

toPrecision() 方法根据最有意义的形式来返回数字的预定形式或指数形式。它有一个参数，即用于表示数的数字总数（不包括指数）。例如，

```
var oNumberObject = new Number(68);
alert(oNumberObject.toPrecision(1));  //输出 "7e+1"
```

这段代码的任务是用一位数字表示数字 68，结果为 "7e+1"，以另外的形式表示即 70。的确，toPrecision() 方法会对数进行舍入。不过，如果用 2 位数字表示 68，就容易多了：

```
var oNumberObject = new Number(68);
alert(oNumberObject.toPrecision(2));  //输出 "68"
```

当然，输出的是 "68"，因为这正是该数的准确表示。不过，如果指定的位数多于需要的位数又如何呢？

```
var oNumberObject = new Number(68);
alert(oNumberObject.toPrecision(3));  //输出 "68.0"
```

在这种情况下，toPrecision(3) 等价于 toFixed(1)，输出的是 "68.0"。

toFixed()、toExponential() 和 toPrecision() 方法都会进行舍入操作，以便用正确的小数位数正确地表示一个数。

提示：与 Boolean 对象相似，Number 对象也很重要，不过应该少用这种对象，以避免潜在的问题。只要可能，都使用数字的原始表示法。

参阅

如需更多有关 Number 对象的信息，请访问 JavaScript Number 对象参考手册。

String 对象

String 对象是 String 原始类型的对象表示法，它是以下方式创建的：

```
var oStringObject = new String("hello world");
```

String 对象的 valueOf() 方法和 toString() 方法都会返回 String 类型的原始值：

```
alert(oStringObject.valueOf() == oStringObject.toString());    //输
```

如果运行这段代码，输出是 "true"，说明这些值真的相等。

注释：String 对象是 ECMAScript 中比较复杂的引用类型之一。同样，本节的重点只是 String 类的基本功能。更多的高级功能请阅读本教程相关的章节，或参阅 JavaScript String 对象参考手册。

length 属性

String 对象具有属性 length，它是字符串中的字符个数：

```
var oStringObject = new String("hello world");
alert(oStringObject.length);    //输出 "11"
```

这个例子输出的是 "11"，即 "hello world" 中的字符个数。注意，即使字符串包含双字节的字符（与 ASCII 字符相对，ASCII 字符只占用一个字节），每个字符也只算一个字符。

charAt() 和 charCodeAt() 方法

String 对象还拥有大量的方法。

首先，两个方法 charAt() 和 charCodeAt() 访问的是字符串中的单个字符。这两个方法都有一个参数，即要操作的字符的位置。

charAt() 方法返回的是包含指定位置处的字符的字符串：

```
var oStringObject = new String("hello world");
alert(oStringObject.charAt(1));    //输出 "e"
```

在字符串 "hello world" 中，位置 1 处的字符是 "e"。在“ECMAScript 原始类型”这一节中我们讲过，第一个字符的位置是 0，第二个字符的位置是 1，依此类推。因此，调用 charAt(1) 返回的是 "e"。

如果想得到的不是字符，而是字符代码，那么可以调用 charCodeAt() 方法：

```
var oStringObject = new String("hello world");
alert(oStringObject.charCodeAt(1));    //输出 "101"
```

这个例子输出 "101"，即小写字母 "e" 的字符代码。

concat() 方法

接下来是 concat() 方法，用于把一个或多个字符串连接到 String 对象的原始值上。该方法返回的是 String 原始值，保持原始的 String 对象不变：

```
var oStringObject = new String("hello ");
var sResult = oStringObject.concat("world");
alert(sResult);        //输出 "hello world"
alert(oStringObject);    //输出 "hello "
```

在上面这段代码中，调用 concat() 方法返回的是 "hello world"，而 String 对象存放的仍然是 "hello "。出于这种原因，较常见的是用加号（+）连接字符串，因为这种形式从逻辑上表明了真正的行为：

```
var oStringObject = new String("hello ");
var sResult = oStringObject + "world";
alert(sResult);        //输出 "hello world"
alert(oStringObject);    //输出 "hello "
```

indexOf() 和 lastIndexOf() 方法

迄今为止，已讨论过连接字符串的方法，访问字符串中的单个字符的方法。不过如果无法确定在某个字符串中是否确实存在一个字符，应该调用什么方法呢？这时，可调用 indexOf() 和 lastIndexOf() 方法。

indexOf() 和 lastIndexOf() 方法返回的都是指定的子串在另一个字符串中的位置，如果没有找不到子串，则返回 -1。

这两个方法的不同之处在于，indexOf() 方法是从字符串的开头（位置 0）开始检索字符串，而 lastIndexOf() 方法则是从字符串的结尾开始检索子串。例如：

```
var oStringObject = new String("hello world!");
alert(oStringObject.indexOf("o"));        输出 "4"
alert(oStringObject.lastIndexOf("o"));    输出 "7"
```

在这里，第一个 "o" 字符串出现在位置 4，即 "hello" 中的 "o"；最后一个 "o" 出现在位置 7，即 "world" 中的 "o"。如果该字符串中只有一个 "o" 字符串，那么 indexOf() 和 lastIndexOf() 方法返回的位置相同。

localeCompare() 方法

下一个方法是 localeCompare()，对字符串进行排序。该方法有一个参数 - 要进行比较的字符串，返回的是下列三个值之一：

- 如果 String 对象按照字母顺序排在参数中的字符串之前，返回负数。
- 如果 String 对象等于参数中的字符串，返回 0
- 如果 String 对象按照字母顺序排在参数中的字符串之后，返回正数。

注释：如果返回负数，那么最常见的是 -1，不过真正返回的是由实现决定的。如果返回正数，那么同样的，最常见的是 1，不过真正返回的是由实现决定的。

示例如下：

```
var oStringObject = new String("yellow");
alert(oStringObject.localeCompare("brick"));        //输出 "1"
alert(oStringObject.localeCompare("yellow"));        //输出 "0"
alert(oStringObject.localeCompare("zoo"));        //输出 "-1"
```

在这段代码中，字符串 "yellow" 与 3 个值进行了对比，即 "brick"、"yellow" 和 "zoo"。由于按照字母顺序排列，"yellow" 位于 "brick" 之后，所以 localeCompare() 返回 1；"yellow" 等于 "yellow"，所以 localeCompare() 返回 0；"zoo" 位于 "yellow" 之后，localeCompare() 返回 -1。再强调一次，由于返回的值是由实现决定的，所以最好以下面的方式调用 localeCompare() 方法：

```
var oStringObject1 = new String("yellow");
var oStringObject2 = new String("brick");

var iResult = oStringObject1.localeCompare(oStringObject2);

if(iResult < 0) {
  alert(oStringObject1 + " comes before " + oStringObject2);
} else if (iResult > 0) {
  alert(oStringObject1 + " comes after " + oStringObject2);
} else {
  alert("The two strings are equal");
}
```

采用这种结构，可以确保这段代码在所有实现中都能正确运行。

localeCompare() 方法的独特之处在于，实现所处的区域（locale，兼指国家/地区和语言）确切说明了这种方法运行的方式。在美国，英语是 ECMAScript 实现的标准语言，localeCompare() 是区分大小写的，大写字母在字母顺序上排在小写字母之后。不过，在其他区域，情况可能并非如此。

slice() 和 substring()

ECMAScript 提供了两种方法从子串创建字符串值，即 slice() 和 substring()。这两种方法返回的都是要处理的字符串的子串，都接受一个或两个参数。第一个参数是要获取的子串的起始位置，第二个参数（如果使用的话）是要获取子串终止前的位置（也就是说，获取终止位置处的字符不包括在返回的值内）。如果省略第二个参数，终止位就默认为字符串的长度。

与 concat() 方法一样，slice() 和 substring() 方法都不改变 String 对象自身的值。它们只返回原始的 String 值，保持 String 对象不变。

```
var oStringObject = new String("hello world");
alert(oStringObject.slice("3"));        //输出 "lo world"
alert(oStringObject.substring("3"));        //输出 "lo world"
alert(oStringObject.slice("3", "7"));        //输出 "lo w"
alert(oStringObject.substring("3", "7"));    //输出 "lo w"
```

在这个例子中，slice() 和 substring() 的用法相同，返回值也一样。当只有参数 3 时，两个方法返回的都是 "lo world"，因为 "hello" 中的第二个 "l" 位于位置 3 上。当有两个参数 "3" 和 "7" 时，两个方法返回的值都是 "lo w"（"world" 中的字母 "o" 位于位置 7 上，所以它不包括在结果中）。

为什么有两个功能完全相同的方法呢？事实上，这两个方法并不完全相同，不过只在参数为负数时，它们处理参数的方式才稍有不同。

对于负数参数，slice() 方法会用字符串的长度加上参数，substring() 方法则将其作为 0 处理（也就是说将忽略它）。例如：

```
var oStringObject = new String("hello world");
alert(oStringObject.slice("-3"));        //输出 "rld"
alert(oStringObject.substring("-3"));    //输出 "hello world"
alert(oStringObject.slice("3, -4"));        //输出 "lo w"
alert(oStringObject.substring("3, -4"));    //输出 "hel"
```

这样即可看出 slice() 和 substring() 方法的主要不同。

当只有参数 -3 时，slice() 返回 "rld"，substring() 则返回 "hello world"。这是因为对于字符串 "hello world"，slice("-3") 将被转换成 slice("8")，而 substring("-3") 将被转换成 substring("0")。

同样，使用参数 3 和 -4 时，差别也很明显。slice() 将被转换成 slice(3, 7)，与前面的例子相同，返回 "lo w"。而 substring() 方法则将两个参数解释为 substring(3, 0)，实际上即 substring(0, 3)，因为 substring() 总把较小的数字作为起始位，较大的数字作为终止位。因此，substring("3, -4") 返回的是 "hel"。这里的最后一行代码用来说明如何使用这些方法。

toLowerCase()、toLocaleLowerCase()、toUpperCase() 和 toLocaleUpperCase()

最后一套要讨论的方法涉及大小写转换。有 4 种方法用于执行大小写转换，即

- toLowerCase()
- toLocaleLowerCase()
- toUpperCase()
- toLocaleUpperCase()

从名字上可以看出它们的用途，前两种方法用于把字符串转换成全小写的，后两种方法用于把字符串转换成全大写的。

toLowerCase() 和 toUpperCase() 方法是原始的，是以 java.lang.String 中相同方法为原型实现的。

toLocaleLowerCase() 和 toLocaleUpperCase() 方法是基于特定的区域实现的（与 localeCompare() 方法相同）。在许多区域中，区域特定的方法都与通用的方法完全相同。不过，有几种语言对 Unicode 大小写转换应用了特定的规则（例如土耳其语），因此必须使用区域特定的方法才能进行正确的转换。

```
var oStringObject = new String("Hello World");
alert(oStringObject.toLocaleUpperCase());    //输出 "HELLO WORLD"
alert(oStringObject.toUpperCase());        //输出 "HELLO WORLD"
alert(oStringObject.toLocaleLowerCase());    //输出 "hello world"
alert(oStringObject.toLowerCase());        //输出 "hello world"
```

这段代码中，toUpperCase() 和 toLocaleUpperCase() 输出的都是 "HELLO WORLD"，toLowerCase() 和 toLocaleLowerCase() 输出的都是 "hello world"。一般来说，如果不知道在以哪种编码运行一种语言，则使用区域特定的方法比较安全。

提示：记住，String 对象的所有属性和方法都可应用于 String 原始值上，因为它们是伪对象。

instanceof 运算符

在使用 typeof 运算符时采用引用类型存储值会出现一个问题，无论引用的是什么类型的对象，它都返回 "object"。ECMAScript 引入了另一个 Java 运算符 instanceof 来解决这个问题。

instanceof 运算符与 typeof 运算符相似，用于识别正在处理的对象的类型。与 typeof 方法不同的是，instanceof 方法要求开发者明确地确认对象为某特定类型。例如：

```
var oStringObject = new String("hello world");
alert(oStringObject instanceof String);    //输出 "true"
```

这段代码问的是"变量 oStringObject 是否为 String 对象的实例？"oStringObject 的确是 String 对象的实例，因此结果是 "true"。尽管不像 typeof 方法那样灵活，但是在 typeof 方法返回 "object" 的情况下，instanceof 方法还是很有用的。

ECMAScript 运算符

ECMAScript 一元运算符

一元运算符只有一个参数，即要操作的对象或值。它们是 **ECMAScript** 中最简单的运算符。

delete

delete 运算符删除对以前定义的对象属性或方法的引用。例如：

```
var o = new Object;
o.name = "David";
alert(o.name);    //输出 "David"
`delete o.name;`
alert(o.name);    //输出 "undefined"
```

在这个例子中，删除了 name 属性，意味着强制解除对它的引用，将其设置为 undefined（即创建的未初始化的变量的值）。

delete 运算符不能删除开发者未定义的属性和方法。例如，下面的代码将引发错误：

```
delete o.toString;
```

即使 toString 是有效的方法名，这行代码也会引发错误，因为 toString() 方法是原始的 ECMAScript 方法，不是开发者定义的。

void

void 运算符对任何值返回 undefined。该运算符通常用于避免输出不应该输出的值，例如，从 HTML 的 <a> 元素调用 JavaScript 函数时。要正确做到这一点，函数不能返回有效值，否则浏览器将清空页面，只显示函数的结果。例如：

```
<a href="javascript:window.open('about:blank')">Click me</a>
```

如果把这行代码放入 HTML 页面，点击其中的链接，即可看到屏幕上显示 "[object]"。TIY

这是因为 window.open() 方法返回了新打开的窗口的引用。然后该对象将被转换成要显示的字符串。

要避免这种效果，可以用 void 运算符调用 window.open() 函数：

```
<a href="javascript:void(window.open('about:blank'))">Click me</a>
```

这使 window.open() 调用返回 undefined，它不是有效值，不会显示在浏览器窗口中。

提示：请记住，没有返回值的函数真正返回的都是 undefined。

前增量/前减量运算符

直接从 C（和 Java）借用的两个运算符是前增量运算符和前减量运算符。

所谓前增量运算符，就是数值上加 1，形式是在变量前放两个加号（++）：

```
var iNum = 10;
++iNum;
```

第二行代码把 iNum 增加到了 11，它实质上等价于：

```
var iNum = 10;
iNum = iNum + 1;
```

同样，前减量运算符是从数值上减 1，形式是在变量前放两个减号（--）：

```
var iNum = 10;
--iNum;
```

在这个例子中，第二行代码把 iNum 的值减到 9。

在使用前缀式运算符时，注意增量和减量运算符都发生在计算表达式之前。考虑下面的例子：

```
var iNum = 10;
--iNum;
alert(iNum);    //输出 "9"
alert(--iNum);    //输出 "8"
alert(iNum);    //输出 "8"
```

第二行代码对 iNum 进行减量运算，第三行代码显示的结果是（"9"）。第四行代码又对 iNum 进行减量运算，不过这次前减量运算和输出操作出现在同一个语句中，显示的结果是 "8"。为了证明已实现了所有的减量操作，第五行代码又输出一次"8"。

在算术表达式中，前增量和前减量运算符的优先级是相同的，因此要按照从左到右的顺序计算之。例如：

```
var iNum1 = 2;
var iNum2 = 20;
var iNum3 = --iNum1 + ++iNum2;    //等于 "22"
var iNum4 = iNum1 + iNum2;        //等于 "22"
```

在前面的代码中，iNum3 等于 22，因为表达式要计算的是 1 + 21。变量 iNum4 也等于 22，也是 1 + 21。

后增量/后减量运算符

还有两个直接从 C（和 Java）借用的运算符，即后增量运算符和后减量运算符。

后增量运算符也是给数值上加 1，形式是在变量后放两个加号（++）：

```
var iNum = 10;
iNum++;
```

不出所料，后减量运算符也是从数值上减 1，形式为在变量后加两个减号（--）：

```
var iNum = 10;
iNum--;
```

第二行代码把 iNum 的 值减到 9。

与前缀式运算符不同的是，后缀式运算符是在计算过包含它们的表达式后才进行增量或减量运算的。考虑以下的例子：

```
var iNum = 10;
iNum--;
alert(iNum);    //输出 "9"
alert(iNum--);    //输出 "9"
alert(iNum);    //输出 "8"
```

与前缀式运算符的例子相似，第二行代码对 iNum 进行减量运算，第三行代码显示结果（"9"）。第四行代码继续显示 iNum 的值，不过这次是在同一语句中应用减量运算符。由于减量运算发生在计算过表达式之后，所以这条语句显示的数是 "9"。执行了第五行代码后，alert 函数显示的是 "8"，因为在执行第四行代码之后和执行第五行代码之前，执行了后减量运算。

在算术表达式中，后增量和后减量运算符的优先级是相同的，因此要按照从左到右的顺序计算之。例如：

```
var iNum1 = 2;
var iNum2 = 20;
var iNum3 = iNum1-- + iNum2++;    //等于 "22"
var iNum4 = iNum1 + iNum2;        //等于 "22"
```

在前面的代码中，iNum3 等于 22，因为表达式要计算的是 2 + 20。变量 iNum4 也等于 22，不过它计算的是 1 + 21，因为增量和减量运算都在给 iNum3 赋值后才发生。

一元加法和一元减法

大多数人都熟悉一元加法和一元减法，它们在 ECMAScript 中的用法与您高中数学中学到的用法相同。

一元加法本质上对数字无任何影响：

```
var iNum = 20;
iNum = +iNum;
alert(iNum);    //输出 "20"
```

这段代码对数字 20 应用了一元加法，返回的还是 20。

尽管一元加法对数字无作用，但对字符串却有有趣的效果，会把字符串转换成数字。

```
var sNum = "20";
alert(typeof sNum);    //输出 "string"
var iNum = +sNum;
alert(typeof iNum);    //输出 "number"
```

这段代码把字符串 "20" 转换成真正的数字。当一元加法运算符对字符串进行操作时，它计算字符串的方式与 parseInt() 相似，主要的不同是只有对以 "0x" 开头的字符串（表示十六进制数字），一元运算符才能把它转换成十进制的值。因此，用一元加法转换 "010"，得到的总是 10，而 "0xB" 将被转换成 11。

另一方面，一元减法就是对数值求负（例如把 20 转换成 -20）：

```
var iNum = 20;
iNum = -iNum;
alert(iNum);    //输出 "-20"
```

与一元加法运算符相似，一元减法运算符也会把字符串转换成近似的数字，此外还会对该值求负。例如：

```
var sNum = "20";
alert(typeof sNum);    //输出 "string"
var iNum = -sNum;
alert(iNum);        //输出 "-20"
alert(typeof iNum);    //输出 "number"
```

在上面的代码中，一元减法运算符将把字符串 "-20" 转换成 -20（一元减法运算符对十六进制和十进制的处理方式与一元加法运算符相似，只是它还会对该值求负）。

ECMAScript 位运算符

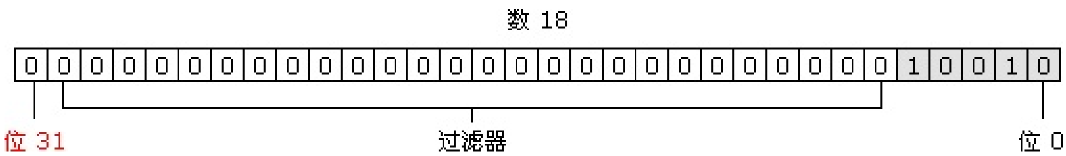
位运算符是在数字底层（即表示数字的 **32** 个数位）进行操作的。

重温整数

ECMAScript 整数有两种类型，即有符号整数（允许用正数和负数）和无符号整数（只允许用正数）。在 ECMAScript 中，所有整数字面量默认都是有符号整数，这意味着什么呢？

有符号整数使用 31 位表示整数的数值，用第 32 位表示整数的符号，0 表示正数，1 表示负数。数值范围从 -2147483648 到 2147483647。

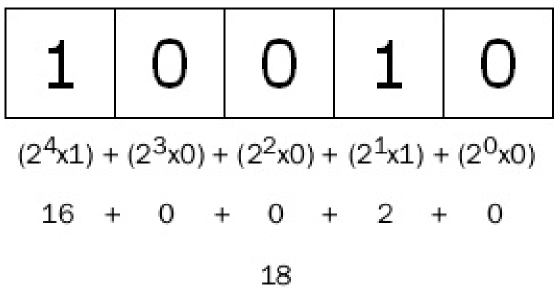
可以以两种不同的方式存储二进制形式的有符号整数，一种用于存储正数，一种用于存储负数。正数是以真二进制形式存储的，前 31 位中的每一位都表示 2 的幂，从第 1 位（位 0）开始，表示 2⁰，第 2 位（位 1）表示 2¹。没用到的位用 0 填充，即忽略不计。例如，下图展示的是数 18 的表示法。



18 的二进制版本只用了前 5 位，它们是这个数字的有效位。把数字转换成二进制字符串，就能看到有效位：

```
var iNum = 18;
alert(iNum.toString(2));    //输出 "10010"
```

这段代码只输出 "10010"，而不是 18 的 32 位表示。其他的数位并不重要，因为仅使用前 5 位即可确定这个十进制数值。如下图所示：



负数也存储为二进制代码，不过采用的形式是二进制补码。计算数字二进制补码的步骤有三步：

1. 确定该数字的非负版本的二进制表示（例如，要计算 -18的二进制补码，首先要确定 18 的二进制表示）

2. 求得二进制反码，即要把 0 替换为 1，把 1 替换为 0
3. 在二进制反码上加 1

要确定 -18 的二进制表示，首先必须得到 18 的二进制表示，如下所示：

```
0000 0000 0000 0000 0000 0000 0001 0010
```

接下来，计算二进制反码，如下所示：

```
1111 1111 1111 1111 1111 1111 1110 1101
```

最后，在二进制反码上加 1，如下所示：

```
1111 1111 1111 1111 1111 1111 1110 1101
                                      1
---------------------------------------
1111 1111 1111 1111 1111 1111 1110 1110
```

因此，-18 的二进制表示即 1111 1111 1111 1111 1111 1111 1110 1110。记住，在处理有符号整数时，开发者不能访问 31 位。

有趣的是，把负整数转换成二进制字符串后，ECMAScript 并不以二进制补码的形式显示，而是用数字绝对值的标准二进制代码前面加负号的形式输出。例如：

```
var iNum = -18;
alert(iNum.toString(2));    //输出 "-10010"
```

这段代码输出的是 "-10010"，而非二进制补码，这是为避免访问位 31。为了简便，ECMAScript 用一种简单的方式处理整数，使得开发者不必关心它们的用法。

另一方面，无符号整数把最后一位作为另一个数位处理。在这种模式中，第 32 位不表示数字的符号，而是值 2³¹。由于这个额外的位，无符号整数的数值范围为 0 到 4294967295。对于小于 2147483647 的整数来说，无符号整数看来与有符号整数一样，而大于 2147483647 的整数则要使用位 31（在有符号整数中，这一位总是 0）。

把无符号整数转换成字符串后，只返回它们的有效位。

注意：所有整数字面量都默认存储为有符号整数。只有 ECMAScript 的位运算符才能创建无符号整数。

位运算 NOT

位运算 NOT 由否定号（~）表示，它是 ECMAScript 中为数不多的与二进制算术有关的运算符之一。

位运算 NOT 是三步的处理过程：

1. 把运算数转换成 32 位数字
2. 把二进制数转换成它的二进制反码
3. 把二进制数转换成浮点数

例如：

```
var iNum1 = 25;        //25 等于 00000000000000000000000000011001
`var iNum2 = ~iNum1;`     //转换为 11111111111111111111111111100110
alert(iNum2);        //输出 "-26"
```

位运算 NOT 实质上是对数字求负，然后减 1，因此 25 变 -26。用下面的方法也可以得到同样的方法：

```
var iNum1 = 25;
var iNum2 = -iNum1 -1;
alert(iNum2);    //输出 -26
```

位运算 AND

位运算 AND 由和号（&）表示，直接对数字的二进制形式进行运算。它把每个数字中的数位对齐，然后用下面的规则对同一位置上的两个数位进行 AND 运算：

第一个数字中的数位	第二个数字中的数位	结果
1	1	1
1	0	0
0	1	0
0	0	0

例如，要对数字 25 和 3 进行 AND 运算，代码如下所示：

```
var iResult = 25 & 3;
alert(iResult);    //输出 "1"
```

25 和 3 进行 AND 运算的结果是 1。为什么？分析如下：

```
 25 = 0000 0000 0000 0000 0000 0000 0001 1001
  3 = 0000 0000 0000 0000 0000 0000 0000 0011
---------------------------------------------
AND = 0000 0000 0000 0000 0000 0000 0000 0001
```

可以看出，在 25 和 3 中，只有一个数位（位 0）存放的都是 1，因此，其他数位生成的都是 0，所以结果为 1。

位运算 OR

位运算 OR 由符号（|）表示，也是直接对数字的二进制形式进行运算。在计算每位时，OR 运算符采用下列规则：

第一个数字中的数位	第二个数字中的数位	结果
1	1	1
1	0	1
0	1	1
0	0	0

仍然使用 AND 运算符所用的例子，对 25 和 3 进行 OR 运算，代码如下：

```
var iResult = 25 | 3;
alert(iResult);    //输出 "27"
```

25 和 3 进行 OR 运算的结果是 27：

```
25 = 0000 0000 0000 0000 0000 0000 0001 1001
 3 = 0000 0000 0000 0000 0000 0000 0000 0011
--------------------------------------------
OR = 0000 0000 0000 0000 0000 0000 0001 1011
```

可以看出，在两个数字中，共有 4 个数位存放的是 1，这些数位被传递给结果。二进制代码 11011 等于 27。

位运算 XOR

位运算 XOR 由符号（^）表示，当然，也是直接对二进制形式进行运算。XOR 不同于 OR，当只有一个数位存放的是 1 时，它才返回 1。真值表如下：

第一个数字中的数位	第二个数字中的数位	结果
1	1	0
1	0	1
0	1	1
0	0	0

对 25 和 3 进行 XOR 运算，代码如下：

```
var iResult = 25 ^ 3;
alert(iResult);    //输出 "26"
```

25 和 3 进行 XOR 运算的结果是 26：

```
 25 = 0000 0000 0000 0000 0000 0000 0001 1001
  3 = 0000 0000 0000 0000 0000 0000 0000 0011
---------------------------------------------
XOR = 0000 0000 0000 0000 0000 0000 0001 1010
```

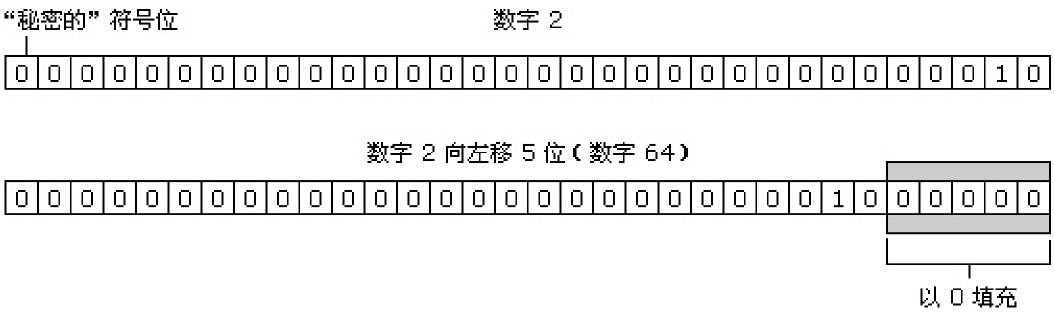
可以看出，在两个数字中，共有 4 个数位存放的是 1，这些数位被传递给结果。二进制代码 11010 等于 26。

左移运算

左移运算由两个小于号表示（<<）。它把数字中的所有数位向左移动指定的数量。例如，把数字 2（等于二进制中的 10）左移 5 位，结果为 64（等于二进制中的 1000000）：

```
var iOld = 2;        //等于二进制 10
var iNew = iOld << 5;    //等于二进制 1000000 十进制 64
```

注意：在左移数位时，数字右边多出 5 个空位。左移运算用 0 填充这些空位，使结果成为完整的 32 位数字。



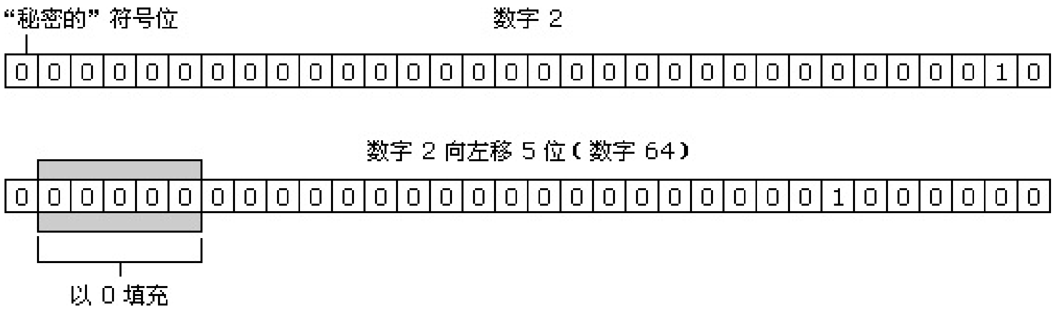
注意：左移运算保留数字的符号位。例如，如果把 -2 左移 5 位，得到的是 -64，而不是 64。“符号仍然存储在第 32 位中吗？”是的，不过这在 ECMAScript 后台进行，开发者不能直接访问第 32 个数位。即使输出二进制字符串形式的负数，显示的也是负号形式（例如，-2 将显示 -10。）

有符号右移运算

有符号右移运算符由两个大于号表示（>>）。它把 32 位数字中的所有数位整体右移，同时保留该数的符号（正号或负号）。有符号右移运算符恰好与左移运算相反。例如，把 64 右移 5 位，将变为 2：

```
var iOld = 64;         //等于二进制 1000000
var iNew = iOld >> 5;    //等于二进制 10 十进制 2
```

同样，移动数位后会造成空位。这次，空位位于数字的左侧，但位于符号位之后。ECMAScript 用符号位的值填充这些空位，创建完整的数字，如下图所示：



无符号右移运算

无符号右移运算符由三个大于号（>>>）表示，它将无符号 32 位数的所有数位整体右移。对于正数，无符号右移运算的结果与有符号右移运算一样。

用有符号右移运算中的例子，把 64 右移 5 位，将变为 2：

```
var iOld = 64;         //等于二进制 1000000
var iNew = iOld >>> 5;    //等于二进制 10 十进制 2
```

对于负数，情况就不同了。

无符号右移运算用 0 填充所有空位。对于正数，这与有符号右移运算的操作一样，而负数则被作为正数来处理。

由于无符号右移运算的结果是一个 32 位的正数，所以负数的无符号右移运算得到的总是一个非常大的数字。例如，如果把 -64 右移 5 位，将得到 134217726。如何得到这种结果的呢？

要实现这一点，需要把这个数字转换成无符号的等价形式（尽管该数字本身还是有符号的），可以通过以下代码获得这种形式：

```
var iUnsigned64 = -64 >>> 0;
```

然后，用 Number 类型的 toString() 获取它的真正的位表示，采用的基为 2：

```
alert(iUnsigned64.toString(2));
```

这将生成 11111111111111111111111111000000，即有符号整数 -64 的二进制补码表示，不过它等于无符号整数 4294967232。

出于这种原因，使用无符号右移运算符要小心。

ECMAScript Boolean 运算符

Boolean 运算符非常重要，它使得程序语言得以正常运行。

Boolean 运算符有三种：**NOT**、**AND** 和 **OR**。

ToBoolean 操作

在学习各种逻辑运算符之前，让我们先了解一下 ECMAScript-262 v5 规范中描述的 ToBoolean 操作。

抽象操作 ToBoolean 将其参数按照下表中的规则转换为逻辑值：

参数类型	结果
Undefined	false
Null	false
Boolean	结果等于输入的参数（不转换）
Number	如果参数为 +0, -0 或 NaN，则结果为 false；否则为 true。
String	如果参数为空字符串，则结果为 false；否则为 true。
Object	true

逻辑 NOT 运算符

在 ECMAScript 中，逻辑 NOT 运算符与 C 和 Java 中的逻辑 NOT 运算符相同，都由感叹号（!）表示。

与逻辑 OR 和逻辑 AND 运算符不同的是，逻辑 **NOT** 运算符返回的一定是 **Boolean** 值。

逻辑 NOT 运算符的行为如下：

- 如果运算数是对象，返回 false
- 如果运算数是数字 0，返回 true
- 如果运算数是 0 以外的任何数字，返回 false
- 如果运算数是 null，返回 true
- 如果运算数是 NaN，返回 true
- 如果运算数是 undefined，发生错误

通常，该运算符用于控制循环：

```
var bFound = false;
var i = 0;

`while (!bFound)` {
  if (aValue[i] == vSearchValues) {
    bFound = true;
  } else {
    i++;
  }
}
```

在这个例子中，Boolean 变量（bFound）用于记录检索是否成功。找到问题中的数据项时，bFound 将被设置为 true，!bFound 将等于 false，意味着运行将跳出 while 循环。

判断 ECMAScript 变量的 Boolean 值时，也可以使用逻辑 NOT 运算符。这样做需要在一行代码中使用两个 NOT 运算符。无论运算数是什么类型，第一个 NOT 运算符返回 Boolean 值。第二个 NOT 将对该 Boolean 值求负，从而给出变量真正的 Boolean 值。

```
var bFalse = false;
var sRed = "red";
var iZero = 0;
var iThreeFourFive = 345;
var oObject = new Object;

document.write("bFalse 的逻辑值是 " + (!!bFalse));
document.write("sRed 的逻辑值是 " + (!!sRed));
document.write("iZero 的逻辑值是 " + (!!iZero));
document.write("iThreeFourFive 的逻辑值是 " + (!!iThreeFourFive));
document.write("oObject 的逻辑值是 " + (!!oObject));
```

结果：

```
bFalse 的逻辑值是 false
sRed 的逻辑值是 true
iZero 的逻辑值是 false
iThreeFourFive 的逻辑值是 true
oObject 的逻辑值是 true
```

逻辑 AND 运算符

在 ECMAScript 中，逻辑 AND 运算符用双和号（&&）表示：

例如：

```
var bTrue = true;
var bFalse = false;
var bResult = bTrue && bFalse;
```

下面的真值表描述了逻辑 AND 运算符的行为：

运算数 1	运算数 2	结果
true	true	true
true	false	false
false	true	false
false	false	false

逻辑 **AND** 运算的运算数可以是任何类型的，不止是 **Boolean** 值。

如果某个运算数不是原始的 Boolean 型值，逻辑 AND 运算并不一定返回 Boolean 值：

- 如果一个运算数是对象，另一个是 Boolean 值，返回该对象。
- 如果两个运算数都是对象，返回第二个对象。
- 如果某个运算数是 null，返回 null。
- 如果某个运算数是 NaN，返回 NaN。
- 如果某个运算数是 undefined，发生错误。

与 Java 中的逻辑 AND 运算相似，ECMAScript 中的逻辑 AND 运算也是简便运算，即如果第一个运算数决定了结果，就不再计算第二个运算数。对于逻辑 AND 运算来说，如果第一个运算数是 false，那么无论第二个运算数的值是什么，结果都不可能等于 true。

考虑下面的例子：

```
var bTrue = true;
var bResult = (bTrue && bUnknown);    //发生错误
alert(bResult);            //这一行不会执行
```

这段代码在进行逻辑 AND 运算时将引发错误，因为变量 bUnknown 是未定义的。变量 bTrue 的值为 true，因为逻辑 AND 运算将继续计算变量 bUnknown。这样做就会引发错误，因为 bUnknown 的值是 undefined，不能用于逻辑 AND 运算。

如果修改这个例子，把第一个数设为 false，那么就不会发生错误：

```
var bFalse = false;
var bResult = (bFalse && bUnknown);
alert(bResult);            //输出 "false"
```

在这段代码中，脚本将输出逻辑 AND 运算返回的值，即字符串 "false"。即使变量 bUnknown 的值为 undefined，它也不会被计算，因为第一个运算数的值是 false。

提示：在使用逻辑 AND 运算符时，必须记住它的这种简便计算特性。

逻辑 OR 运算符

ECMAScript 中的逻辑 OR 运算符与 Java 中的相同，都由双竖线（||）表示：

```
var bTrue = true;
var bFalse = false;
var bResult = bTrue || bFalse;
```

下面的真值表描述了逻辑 OR 运算符的行为：

运算数 1	运算数 2	结果
true	true	true
true	false	true
false	true	true
false	false	false

与逻辑 AND 运算符相似，如果某个运算数不是 Boolean 值，逻辑 OR 运算并不一定返回 Boolean 值：

- 如果一个运算数是对象，并且该对象左边的运算数值均为 false，则返回该对象。
- 如果两个运算数都是对象，返回第一个对象。
- 如果最后一个运算数是 null，并且其他运算数值均为 false，则返回 null。
- 如果最后一个运算数是 NaN，并且其他运算数值均为 false，则返回 NaN。
- 如果某个运算数是 undefined，发生错误。

与逻辑 AND 运算符一样，逻辑 OR 运算也是简便运算。对于逻辑 OR 运算符来说，如果第一个运算数值为 true，就不再计算第二个运算数。

例如：

```
var bTrue = true;
var bResult = (bTrue || bUnknown);
alert(bResult);            //输出 "true"
```

与前面的例子相同，变量 bUnknown 是未定义的。不过，由于变量 bTrue 的值为 true，bUnknown 不会被计算，因此输出的是 "true"。

如果把 bTrue 改为 false，将发生错误：

```
var bFalse = false;
var bResult = (bFalse || bUnknown);    //发生错误
alert(bResult);             //不会执行这一行
```

ECMAScript 乘性运算符

ECMAScript 的乘性运算符与 **Java**、**C**、**Perl** 等于语言中的同类运算符的运算方式相似。

需要注意的是，乘性运算符还具有一些自动转换功能。

乘法运算符

乘法运算符由星号（*）表示，用于两数相乘。

ECMAScript 中的乘法语法与 C 语言中的相同：

```
var iResult = 12 * 34
```

不过，在处理特殊值时，ECMAScript 中的乘法还有一些特殊行为：

- 如果结果太大或太小，那么生成的结果是 Infinity 或 -Infinity。
- 如果某个运算数是 NaN，结果为 NaN。
- Infinity 乘以 0，结果为 NaN。
- Infinity 乘以 0 以外的任何数字，结果为 Infinity 或 -Infinity。
- Infinity 乘以 Infinity，结果为 Infinity。

注释：如果运算数是数字，那么执行常规的乘法运算，即两个正数或两个负数为正数，两个运算数符号不同，结果为负数。

除法运算符

除法运算符由斜杠（/）表示，用第二个运算数除第一个运算数：

```
var iResult = 88 /11;
```

与乘法运算符相似，在处理特殊值时，除法运算符也有一些特殊行为：

- 如果结果太大或太小，那么生成的结果是 Infinity 或 -Infinity。
- 如果某个运算数是 NaN，结果为 NaN。
- Infinity 被 Infinity 除，结果为 NaN。
- Infinity 被任何数字除，结果为 Infinity。
- 0 除一个任何非无穷大的数字，结果为 NaN。
- Infinity 被 0 以外的任何数字除，结果为 Infinity 或 -Infinity。

注释：如果运算数是数字，那么执行常规的除法运算，即两个正数或两个负数为正数，两个运算数符号不同，结果为负数。

取模运算符

除法（余数）运算符由百分号（%）表示，使用方法如下：

```
var iResult = 26%5; //等于 1
```

与其他乘性运算符相似，对于特殊值，取模运算符也有特殊的行为：

- 如果被除数是 Infinity，或除数是 0，结果为 NaN。
- Infinity 被 Infinity 除，结果为 NaN。
- 如果除数是无穷大的数，结果为被除数。
- 如果被除数为 0，结果为 0。

注释：如果运算数是数字，那么执行常规的算术除法运算，返回除法运算得到的余数。

ECMAScript 加性运算符

在多数程序设计语言中，加性运算符（即加号或减号）通常是最简单的数学运算符。

在 **ECMAScript** 中，加性运算符有大量的特殊行为。

加法运算符

法运算符由加号（+）表示：

```
var iResult = 1 + 2
```

与乘性运算符一样，在处理特殊值时，ECMAScript 中的加法也有一些特殊行为：

- 某个运算数是 NaN，那么结果为 NaN。
- -Infinity 加 -Infinity，结果为 -Infinity。
- Infinity 加 -Infinity，结果为 NaN。
- +0 加 +0，结果为 +0。
- -0 加 +0，结果为 +0。
- -0 加 -0，结果为 -0。

不过，如果某个运算数是字符串，那么采用下列规则：

- 如果两个运算数都是字符串，把第二个字符串连接到第一个上。
- 如果只有一个运算数是字符串，把另一个运算数转换成字符串，结果是两个字符串连接成的字符串。

例如：

```
var result = 5 + 5;    //两个数字
alert(result);        //输出 "10"
var result2 = 5 + "5";    //一个数字和一个字符串
alert(result);        //输出 "55"
```

这段代码说明了加法运算符的两种模式之间的差别。正常情况下，5+5 等于 10（原始数值），如上述代码中前两行所示。不过，如果把一个运算数改为字符串 "5"，那么结果将变为 "55"（原始的字符串值），因为另一个运算数也会被转换为字符串。

注意：为了避免 JavaScript 中的一种常见错误，在使用加法运算符时，一定要仔细检查运算数的数据类型。

减法运算符

减法运算符（-），也是一个常用的运算符：

```
var iResult = 2 - 1;
```

与加法运算符一样，在处理特殊值时，减法运算符也有一些特殊行为：

- 某个运算数是 NaN，那么结果为 NaN。
- Infinity 减 Infinity，结果为 NaN。
- -Infinity 减 -Infinity，结果为 NaN。
- Infinity 减 -Infinity，结果为 Infinity。
- -Infinity 减 Infinity，结果为 -Infinity。
- +0 减 +0，结果为 +0。
- -0 减 -0，结果为 -0。
- +0 减 -0，结果为 +0。
- 某个运算符不是数字，那么结果为 NaN。

注释：如果运算数都是数字，那么执行常规的减法运算，并返回结果。

ECMAScript 关系运算符

关系运算符执行的是比较运算。每个关系运算符都返回一个布尔值。

常规比较方式

关系运算符小于、大于、小于等于和大于等于执行的是两个数的比较运算，比较方式与算术比较运算相同。

每个关系运算符都返回一个布尔值：

```
var bResult1 = 2 > 1    //true
var bResult2 = 2 < 1    //false
```

不过，对两个字符串应用关系运算符，它们的行为则不同。许多人认为小于表示“在字母顺序上靠前”，大于表示“在字母顺序上靠后”，但事实并非如此。对于字符串，第一个字符串中每个字符的代码都与会第二个字符串中对应位置的字符的代码进行数值比较。完成这种比较操作后，返回一个 Boolean 值。问题在于大写字母的代码都小于小写字母的代码，这意味这着可能会遇到下列情况：

```
var bResult = "Blue" < "alpha";
alert(bResult);    //输出 true
```

在上面的例子中，字符串 "Blue" 小于 "alpha"，因为字母 B 的字符代码是 66，字母 a 的字符代码是 97。要强制性得到按照真正的字母顺序比较的结果，必须把两个数转换成相同的大小写形式（全大写或全小写的），然后再进行比较：

```
var bResult = "Blue".toLowerCase() < "alpha".toLowerCase();
alert(bResult);    //输出 false
```

把两个运算数都转换成小写，确保了正确识别出 "alpha" 在字母顺序上位于 "Blue" 之前。

比较数字和字符串

另一种棘手的状况发生在比较两个字符串形式的数字时，比如：

```
var bResult = "25" < "3";
alert(bResult);    //输出 "true"
```

上面这段代码比较的是字符串 "25" 和 "3"。两个运算数都是字符串，所以比较的是它们的字符代码（"2" 的字符代码是 50，"3" 的字符代码是 51）。

不过，如果把某个运算数该为数字，那么结果就有趣了：

```
var bResult = "25" < 3;
alert(bResult);    //输出 "false"
```

这里，字符串 "25" 将被转换成数字 25，然后与数字 3 进行比较，结果不出所料。

无论何时比较一个数字和一个字符串，ECMAScript 都会把字符串转换成数字，然后按照数字顺序比较它们。

不过，如果字符串不能转换成数字又该如何呢？考虑下面的例子：

```
var bResult = "a" < 3;
alert(bResult);
```

你能预料到这段代码输出什么吗？字母 "a" 不能转换成有意义的数字。不过，如果对它调用 parseInt() 方法，返回的是 NaN。根据规则，任何包含 NaN 的关系运算符都要返回 false，因此这段代码也输出 false：

```
var bResult = "a" >= 3;
alert(bResult);
```

通常，如果小于运算的两个值返回 false，那么大于等于运算必须返回 true，不过如果某个数字是 NaN，情况则非如此。

ECMAScript 等性运算符

判断两个变量是否相等是程序设计中非常重要的运算。在处理原始值时，这种运算相当简单，但涉及对象，任务就稍有点复杂。

ECMAScript 提供了两套等性运算符：等号和非等号用于处理原始值，全等号和非全等号用于处理对象。

等号和非等号

在 ECMAScript 中，等号由双等号（==）表示，当且仅当两个运算数相等时，它返回 true。非等号由感叹号加等号（!=）表示，当且仅当两个运算数不相等时，它返回 true。为确定两个运算数是否相等，这两个运算符都会进行类型转换。

执行类型转换的规则如下：

- 如果一个运算数是 Boolean 值，在检查相等性之前，把它转换成数字值。false 转换成 0，true 为 1。
- 如果一个运算数是字符串，另一个是数字，在检查相等性之前，要尝试把字符串转换成数字。
- 如果一个运算数是对象，另一个是字符串，在检查相等性之前，要尝试把对象转换成字符串。
- 如果一个运算数是对象，另一个是数字，在检查相等性之前，要尝试把对象转换成数字。

在比较时，该运算符还遵守下列规则：

- 值 null 和 undefined 相等。
- 在检查相等性时，不能把 null 和 undefined 转换成其他值。
- 如果某个运算数是 NaN，等号将返回 false，非等号将返回 true。
- 如果两个运算数都是对象，那么比较的是它们的引用值。如果两个运算数指向同一对象，那么等号返回 true，否则两个运算数不等。

重要提示：即使两个数都是 NaN，等号仍然返回 false，因为根据规则，NaN 不等于 NaN。

下表列出了一些特殊情况，以及它们的结果：

表达式	值
null == undefined	true
"NaN" == NaN	false
5 == NaN	false
NaN == NaN	false
NaN != NaN	true
false == 0	true
true == 1	true
true == 2	false
undefined == 0	false
null == 0	false
"5" == 5	true

全等号和非全等号

等号和非等号的同类运算符是全等号和非全等号。这两个运算符所做的与等号和非等号相同，只是它们在检查相等性前，不执行类型转换。

全等号由三个等号表示（===），只有在无需类型转换运算数就相等的情况下，才返回 true。

例如：

```
var sNum = "66";
var iNum = 66;
alert(sNum == iNum);    //输出 "true"
alert(sNum === iNum);    //输出 "false"
```

在这段代码中，第一个 alert 使用等号来比较字符串 "66" 和数字 66，输出 "true"。如前所述，这是因为字符串 "66" 将被转换成数字 66，，然后才与另一个数字 66 进行比较。第二个 alert 使用全等号在没有类型转换的情况下比较字符串和数字，当然，字符串不等于数字，所以输出 "false"。

非全等号由感叹号加两个等号（!==）表示，只有在无需类型转换运算数不相等的情况下，才返回 true。

例如：

```
var sNum = "66";
var iNum = 66;
alert(sNum != iNum);    //输出 "false"
alert(sNum !== iNum);    //输出 "true"
```

这里，第一个 alert 使用非等号，把字符串 "66" 转换成数字 66，使得它与第二个运算数 66 相等。因此，计算结果为 "false"，因为两个运算数是相等的。第二个 alert 使用的非全等号。该运算是在问："sNum" 与 "iNum" 不同吗？这个问题的答案是：是的（true），因为 sNum 是字符串，而 iNum 是数字，它们当然不同。

ECMAScript 条件运算符

条件运算符

条件运算符是 ECMAScript 中功能最多的运算符，它的形式与 Java 中的相同。

```
_variable = boolean_expression ? true_value : false_value_;
```

该表达式主要是根据 *boolean_expression* 的计算结果有条件地为变量赋值。如果 *Boolean_expression* 为 true，就把 *true_value* 赋给变量；如果它是 false，就把 *false_value* 赋给变量。

例如：

```
var iMax = (iNum1 > iNum2) ? iNum1 : iNum2;
```

在这里例子中，iMax 将被赋予数字中的最大值。表达式声明如果 iNum1 大于 iNum2，则把 iNum1 赋予 iMax。但如果表达式为 false（即 iNum2 大于或等于 iNum1），则把 iNum2 赋予 iMax。

ECMAScript 赋值运算符

赋值运算符

简单的赋值运算符由等号（=）实现，只是把等号右边的值赋予等号左边的变量。

例如：

```
var iNum = 10;
```

复合赋值运算是由乘性运算符、加性运算符或位移运算符加等号（=）实现的。这些赋值运算符是下列这些常见情况的缩写形式：

```
var iNum = 10;
iNum = iNum + 10;
```

可以用一个复合赋值运算符改写第二行代码：

```
var iNum = 10;
iNum += 10;
```

每种主要的算术运算以及其他几个运算都有复合赋值运算符：

- 乘法/赋值（ `*=` ）
- 除法/赋值（ `/=` ）
- 取模/赋值（ `%=` ）
- 加法/赋值（ `+=` ）
- 减法/赋值（ `-=` ）
- 左移/赋值（ `<<=` ）
- 有符号右移/赋值（ `>>=` ）
- 无符号右移/赋值（ `>>>=` ）

ECMAScript 逗号运算符

逗号运算符

用逗号运算符可以在一条语句中执行多个运算。

例如：

```
var iNum1 = 1, iNum = 2, iNum3 = 3;
```

逗号运算符常用变量声明中。

ECMAScript 语句

ECMAScript if 语句

if 语句是 **ECMAScript** 中最常用的语句之一。

ECMAScript 语句

ECMA - 262 描述了 ECMAScript 的几种语句（statement）。

语句主要定义了 ECMAScript 的大部分语句，通常采用一个或多个关键字，完成给定的任务。

语句可以非常简单，例如通知函数退出，也可以非常复杂，如声明一组要反复执行的命令。

在《ECMAScript 语句》这一章，我们介绍了所有标准的 ECMAScript 语句。

if 语句

if 语句是 ECMAScript 中最常用的语句之一，事实上在许多计算机语言中都是如此。

if 语句的语法：

```
`if` (_condition_) _statement1_ `else` _statement2_
```

其中 *condition* 可以是任何表达式，计算的结果甚至不必是真正的 boolean 值，ECMAScript 会把它转换成 boolean 值。

如果条件计算结果为 true，则执行 *statement1*；如果条件计算结果为 false，则执行 *statement2*。

每个语句都可以是单行代码，也可以是代码块。

例如：

```
if (i > 30)
  {alert("大于 30");}
else
  {alert("小于等于 30");}
```

提示：使用代码块被认为是一种最佳的编程实践，即使要执行的代码只有一行。这样做可以使每个条件要执行什么一目了然。

还可以串联多个 if 语句。就像这样：

```
`if` (_condition1_) _statement1_ `else if` (_condition2_) _statemen
```

例如：

```
if (i > 30) {
  alert("大于 30");
} else if (i < 0) {
  alert("小于 0");
} else {
  alert("在 0 到 30 之间");
}
```

ECMAScript 迭代语句

迭代语句又叫循环语句，声明一组要反复执行的命令，直到满足某些条件为止。

循环通常用于迭代数组的值（因此而得名），或者执行重复的算术任务。

本节为您介绍 **ECMAScript** 提供的四种迭代语句。

do-while 语句

do-while 语句是后测试循环，即退出条件在执行循环内部的代码之后计算。这意味着在计算表达式之前，至少会执行循环主体一次。

它的语法如下：

```
`do` {_statement_} `while` (_expression_);
```

例子：

```
var i = 0;
do {i += 2;} while (i < 10);
```

while 语句

while 语句是前测试循环。这意味着退出条件是在执行循环内部的代码之前计算的。因此，循环主体可能根本不被执行。

它的语法如下：

```
`while` (_expression_) _statement_
```

例子：

```
var i = 0;
while (i < 10) {
  i += 2;
}
```

for 语句

for 语句是前测试循环，而且在进入循环之前，能够初始化变量，并定义循环后要执行的代码。

它的语法如下：

```
`for` (_initialization; expression; post-loop-expression_) _stateme
```

注意：*post-loop-expression* 之后不能写分号，否则无法运行。

例子：

```
iCount = 6;
for (var i = 0; i < iCount; i++) {
  alert(i);
}
```

这段代码定义了初始值为 0 的变量 i。只有当条件表达式（i < iCount）的值为 true 时，才进入 for 循环，这样循环主体可能不被执行。如果执行了循环主体，那么将执行循环后表达式，并迭代变量 i。

for-in 语句

for 语句是严格的迭代语句，用于枚举对象的属性。

它的语法如下：

```
`for` (_property_ `in` _expression_) _statement_
```

例子：

```
for (sProp in window) {
  alert(sProp);
}
```

这里，for-in 语句用于显示 window 对象的所有属性。

前面讨论过的 PropertyIsEnumerable() 是 ECMAScript 中专门用于说明属性是否可以用 for-in 语句访问的方法。

ECMAScript 标签语句

有标签的语句

可以用下列语句给语句加标签，以便以后调用：

```
_label_ : _statement_
```

例如：

```
start : i = 5;
```

在这个例子中，标签 start 可以被之后的 break 或 continue 语句引用。

提示：在下面的章节，我们将为您介绍 break 和 continue 语句。

ECMAScript break 和 continue 语句

break 和 continue 语句对循环中的代码执行提供了更严格的控制。

break 和 continue 语句的不同之处

break 语句可以立即退出循环，阻止再次反复执行任何代码。

而 continue 语句只是退出当前循环，根据控制表达式还允许继续进行下一次循环。

例如：

```
var iNum = 0;

for (var i=1; i<10; i++) {
  if (i % 5 == 0) {
    `break;`
  }
  iNum++;
}
alert(iNum);    //输出 "4"
```

在以上代码中，for 循环从 1 到 10 迭代变量 i。在循环主体中，if 语句将（使用取模运算符）检查 i 的值是否能被 5 整除。如果能被 5 整除，将执行 break 语句。alert 显示 "4"，即退出循环前执行循环的次数。

如果用 continue 语句代替这个例子中的 break 语句，结果将不同：

```
var iNum = 0;

for (var i=1; i<10; i++) {
  if (i % 5 == 0) {
    `continue;`
  }
  iNum++;
}
alert(iNum);    //输出 "8"
```

这里，alert 将显示 "8"，即执行循环的次数。可能执行的循环总数为 9，不过当 i 的值为 5 时，将执行 continue 语句，会使循环跳过表达式 iNum++，返回循环开头。

与有标签的语句一起使用

break 语句和 continue 语句都可以与有标签的语句联合使用，返回代码中的特定位置。

通常，当循环内部还有循环时，会这样做，例如：

```
var iNum = 0;

outermost:
for (var i=0; i<10; i++) {
  for (var j=0; j<10; j++) {
    if (i == 5 && j == 5) {
    `break outermost;`
  }
  iNum++;
  }
}

alert(iNum);    //输出 "55"
```

在上面的例子中，标签 outermost 表示的是第一个 for 语句。正常情况下，每个 for 语句执行 10 次代码块，这意味着 iNum++ 正常情况下将被执行 100 次，在执行完成时，iNum 应该等于 100。这里的 break 语句有一个参数，即停止循环后要跳转到的语句的标签。这样 break 语句不止能跳出内部 for 语句（即使用变量 j 的语句），还能跳出外部 for 语句（即使用变量 i 的语句）。因此，iNum 最后的值是 55，因为当 i 和 j 的值都等于 5 时，循环将终止。

可以以相同的方式使用 continue 语句：

```
var iNum = 0;

outermost:
for (var i=0; i<10; i++) {
  for (var j=0; j<10; j++) {
    if (i == 5 && j == 5) {
    `continue outermost;`
  }
  iNum++;
  }
}

alert(iNum);    //输出 "95"
```

在上例中，continue 语句会迫使循环继续，不止是内部循环，外部循环也如此。当 j 等于 5 时出现这种情况，意味着内部循环将减少 5 次迭代，致使 iNum 的值为 95。

提示：可以看出，与 break 和 continue 联合使用的有标签语句非常强大，不过过度使用它们会给调试代码带来麻烦。要确保使用的标签具有说明性，同时不要嵌套太多层循环。

提示：想了解什么是有标签语句，请阅读 ECMAScript 标签语句 这一节。

ECMAScript with 语句

有标签的语句

with 语句用于设置代码在特定对象中的作用域。

它的语法：

```
with (_expression_) _statement_
```

例如：

```
var sMessage = "hello";
with(sMessage) {
  alert(toUpperCase());    //输出 "HELLO"
}
```

在这个例子中，with 语句用于字符串，所以在调用 toUpperCase() 方法时，解释程序将检查该方法是否是本地函数。如果不是，它将检查伪对象 sMessage，看它是否为该对象的方法。然后，alert 输出 "HELLO"，因为解释程序找到了字符串 "hello" 的 toUpperCase() 方法。

提示：with 语句是运行缓慢的代码块，尤其是在已设置了属性值时。大多数情况下，如果可能，最好避免使用它。

ECMAScript switch 语句

switch 语句

switch 语句是 if 语句的兄弟语句。

开发者可以用 switch 语句为表达式提供一系列的情况（case）。

switch 语句的语法：

```
switch (_expression_)
  case _value_: _statement_;
    break;
  case _value_: _statement_;
    break;
  case _value_: _statement_;
    break;
  case _value_: _statement_;
    break;
...
  case _value_: _statement_;
    break;
  default: _statement_;
```

每个情况（case）都是表示“如果 *expression* 等于 *value*，就执行 *statement*”。

关键字 break 会使代码跳出 switch 语句。如果没有关键字 break，代码执行就会继续进入下一个 case。

关键字 default 说明了表达式的结果不等于任何一种情况时的操作（事实上，它相对于 else 从句）。

switch 语句主要是为避免让开发者编写下面的代码：

```
if (i == 20)
  alert("20");
else if (i == 30)
  alert("30");
else if (i == 40)
  alert("40");
else
  alert("other");
```

等价的 switch 语句是这样的：

```
switch (i) {
  case 20: alert("20");
    break;
  case 30: alert("30");
    break;
  case 40: alert("40");
    break;
  default: alert("other");
}
```

ECMAScript 和 Java 中的 switch 语句

ECMAScript 和 Java 中的 switch 语句有两点不同。在 ECMAScript 中，switch 语句可以用于字符串，而且能用不是常量的值说明情况：

```
var BLUE = "blue", RED = "red", GREEN  = "green";

switch (sColor) {
  case BLUE: alert("Blue");
    break;
  case RED: alert("Red");
    break;
  case GREEN: alert("Green");
    break;
  default: alert("Other");
}
```

这里，switch 语句用于字符串 sColor，声明 case 使用的是变量 BLUE、RED 和 GREEN，这在 ECMAScript 中是完全有效的。

ECMAScript 函数

ECMAScript 函数概述

什么是函数？

函数是一组可以随时随地运行的语句。

函数是 ECMAScript 的核心。

函数是由这样的方式进行声明的：关键字 function、函数名、一组参数，以及置于括号中的待执行代码。

函数的基本语法是这样的：

```
function _functionName(arg0, arg1, ... argN)_ {
  _statements_
}
```

例如：

```
function sayHi(sName, sMessage) {
  alert("Hello " + sName + sMessage);
}
```

如何调用函数？

函数可以通过其名字加上括号中的参数进行调用，如果有多个参数。

如果您想调用上例中的那个函数，可以使用如下的代码：

```
sayHi("David", " Nice to meet you!")
```

调用上面的函数 sayHi() 会生成一个警告窗口。您可以亲自试一试这个例子。

函数如何返回值？

函数 sayHi() 未返回值，不过不必专门声明它（像在 Java 中使用 void 那样）。

即使函数确实有值，也不必明确地声明它。该函数只需要使用 return 运算符后跟要返回的值即可。

```
function sum(iNum1, iNum2) {
  return iNum1 + iNum2;
}
```

下面的代码把 sum 函数返回的值赋予一个变量：

```
var iResult = sum(1,1);
alert(iResult);    //输出 "2"
```

另一个重要概念是，与在 Java 中一样，函数在执行过 return 语句后立即停止代码。因此，return 语句后的代码都不会被执行。

例如，在下面的代码中，alert 窗口就不会显示出来：

```
function sum(iNum1, iNum2) {
  return iNum1 + iNum2;
  alert(iNum1 + iNum2);
}
```

一个函数中可以有多个 return 语句，如下所示：

```
function diff(iNum1, iNum2) {
  if (iNum1 > iNum2) {
    return iNum1 - iNum2;
  } else {
    return iNum2 - iNum1;
  }
}
```

上面的函数用于返回两个数的差。要实现这一点，必须用较大的数减去较小的数，因此用 if 语句决定执行哪个 return 语句。

如果函数无返回值，那么可以调用没有参数的 return 运算符，随时退出函数。

例如：

```
function sayHi(sMessage) {
  if (sMessage == "bye") {
    return;
  }

  alert(sMessage);
}
```

这段代码中，如果 sMessage 等于 "bye"，就永远不显示警告框。

注释：如果函数无明确的返回值，或调用了没有参数的 return 语句，那么它真正返回的值是 undefined。

ECMAScript arguments 对象

arguments 对象

在函数代码中，使用特殊对象 arguments，开发者无需明确指出参数名，就能访问它们。

例如，在函数 sayHi() 中，第一个参数是 message。用 arguments[0] 也可以访问这个值，即第一个参数的值（第一个参数位于位置 0，第二个参数位于位置 1，依此类推）。

因此，无需明确命名参数，就可以重写函数：

```
function sayHi() {
  if (`arguments[0]` == "bye") {
    return;
  }

  alert(arguments[0]);
}
```

检测参数个数

还可以用 arguments 对象检测函数的参数个数，引用属性 arguments.length 即可。

下面的代码将输出每次调用函数使用的参数个数：

```
function howManyArgs() {
  alert(arguments.length);
}

howManyArgs("string", 45);
howManyArgs();
howManyArgs(12);
```

上面这段代码将依次显示 "2"、"0" 和 "1"。

注释：与其他程序设计语言不同，ECMAScript 不会验证传递给函数的参数个数是否等于函数定义的参数个数。开发者定义的函数都可以接受任意个数的参数（根据 Netscape 的文档，最多可接受 255 个），而不会引发任何错误。任何遗漏的参数都会以 undefined 传递给函数，多余的函数将忽略。

模拟函数重载

用 arguments 对象判断传递给函数的参数个数，即可模拟函数重载：

```
function doAdd() {
  if(arguments.length == 1) {
    alert(arguments[0] + 5);
  } else if(arguments.length == 2) {
    alert(arguments[0] + arguments[1]);
  }
}

doAdd(10);    //输出 "15"
doAdd(40, 20);    //输出 "60"
```

当只有一个参数时，doAdd() 函数给参数加 5。如果有两个参数，则会把两个参数相加，返回它们的和。所以，doAdd(10) 输出的是 "15"，而 doAdd(40, 20) 输出的是 "60"。

虽然不如重载那么好，不过已足以避开 ECMAScript 的这种限制。

ECMAScript Function 对象（类）

ECMAScript 的函数实际上是功能完整的对象。

Function 对象（类）

ECMAScript 最令人感兴趣的可能莫过于函数实际上是功能完整的对象。

Function 类可以表示开发者定义的任何函数。

用 Function 类直接创建函数的语法如下：

```
var function_name = new function(_arg1_, _arg2_, ..., _argN_, _func
```

在上面的形式中，每个 *arg* 都是一个参数，最后一个参数是函数主体（要执行的代码）。这些参数必须是字符串。

记得下面这个函数吗？

```
function sayHi(sName, sMessage) {
  alert("Hello " + sName + sMessage);
}
```

还可以这样定义它：

```
var sayHi
=
new Function("sName", "sMessage", "alert(\"Hello \" + sName + sMess
```

虽然由于字符串的关系，这种形式写起来有些困难，但有助于理解函数只不过是一种引用类型，它们的行为与用 Function 类明确创建的函数行为是相同的。

请看下面这个例子：

```
function doAdd(iNum) {
  alert(iNum + 20);
}

function doAdd(iNum) {
  alert(iNum + 10);
}

doAdd(10);    //输出 "20"
```

如你所知，第二个函数重载了第一个函数，使 doAdd(10) 输出了 "20"，而不是 "30"。

如果以下面的形式重写该代码块，这个概念就清楚了：

```
var doAdd = new Function("iNum", "alert(iNum + 20)");
var doAdd = new Function("iNum", "alert(iNum + 10)");
doAdd(10);
```

请观察这段代码，很显然，doAdd 的值被改成了指向不同对象的指针。函数名只是指向函数对象的引用值，行为就像其他对象一样。甚至可以使两个变量指向同一个函数：

```
var doAdd = new Function("iNum", "alert(iNum + 10)");
var alsodoAdd = doAdd;
doAdd(10);    //输出 "20"
alsodoAdd(10);    //输出 "20"
```

在这里，变量 doAdd 被定义为函数，然后 alsodoAdd 被声明为指向同一个函数的指针。用这两个变量都可以执行该函数的代码，并输出相同的结果 - "20"。因此，如果函数名只是指向函数的变量，那么可以把函数作为参数传递给另一个函数吗？回答是肯定的！

```
function callAnotherFunc(fnFunction, vArgument) {
  fnFunction(vArgument);
}

var doAdd = new Function("iNum", "alert(iNum + 10)");

callAnotherFunc(doAdd, 10);    //输出 "20"
```

在上面的例子中，callAnotherFunc() 有两个参数 - 要调用的函数和传递给该函数的参数。这段代码把 doAdd() 传递给 callAnotherFunc() 函数，参数是 10，输出 "20"。

注意：尽管可以使用 Function 构造函数创建函数，但最好不要使用它，因为用它定义函数比用传统方式要慢得多。不过，所有函数都应看作 Function 类的实例。

Function 对象的 length 属性

如前所述，函数属于引用类型，所以它们也有属性和方法。

ECMAScript 定义的属性 length 声明了函数期望的参数个数。例如：

```
function doAdd(iNum) {
  alert(iNum + 10);
}

function sayHi() {
  alert("Hi");
}

alert(doAdd.length);    //输出 "1"
alert(sayHi.length);    //输出 "0"
```

函数 doAdd() 定义了一个参数，因此它的 length 是 1；sayHi() 没有定义参数，所以 length 是 0。

记住，无论定义了几个参数，ECMAScript 可以接受任意多个参数（最多 25 个），这一点在《函数概述》这一章中讲解过。属性 length 只是为查看默认情况下预期的参数个数提供了一种简便方式。

Function 对象的方法

Function 对象也有与所有对象共享的 valueOf() 方法和 toString() 方法。这两个方法返回的都是函数的源代码，在调试时尤其有用。例如：

```
function doAdd(iNum) {
  alert(iNum + 10);
}

document.write(doAdd.toString());
```

上面这段代码输出了 doAdd() 函数的文本。！

ECMAScript 闭包（closure）

ECMAScript 最易让人误解的一点是，它支持闭包（**closure**）。

闭包，指的是词法表示包括不被计算的变量的函数，也就是说，函数可以使用函数之外定义的变量。

简单的闭包实例

在 ECMAScript 中使用全局变量是一个简单的闭包实例。请思考下面这段代码：

```
var sMessage = "hello world";

function sayHelloWorld() {
  alert(sMessage);
}

sayHelloWorld();
```

在上面这段代码中，脚本被载入内存后，并没有为函数 sayHelloWorld() 计算变量 sMessage 的值。该函数捕获 sMessage 的值只是为了以后的使用，也就是说，解释程序知道在调用该函数时要检查 sMessage 的值。sMessage 将在函数调用 sayHelloWorld() 时（最后一行）被赋值，显示消息 "hello world"。

复杂的闭包实例

在一个函数中定义另一个会使闭包变得更加复杂。例如：

```
var iBaseNum = 10;

function addNum(iNum1, iNum2) {
  function doAdd() {
    return iNum1 + iNum2 + iBaseNum;
  }
  return doAdd();
}
```

这里，函数 addNum() 包括函数 doAdd() （闭包）。内部函数是一个闭包，因为它将获取外部函数的参数 iNum1 和 iNum2 以及全局变量 iBaseNum 的值。addNum() 的最后一步调用了 doAdd()，把两个参数和全局变量相加，并返回它们的和。

这里要掌握的重要概念是，doAdd() 函数根本不接受参数，它使用的值是从执行环境中获取的。

可以看到，闭包是 ECMAScript 中非常强大多用的一部分，可用于执行复杂的计算。

提示：就像使用任何高级函数一样，使用闭包要小心，因为它们可能会变得非常复杂。

ECMAScript 对象

ECMAScript 面向对象技术

面向对象术语

对象

ECMA-262 把对象（object）定义为“属性的无序集合，每个属性存放一个原始值、对象或函数”。严格来说，这意味着对象是无特定顺序的值的数组。

尽管 ECMAScript 如此定义对象，但它更通用的定义是基于代码的名词（人、地点或事物）的表示。

类

每个对象都由类定义，可以把类看做对象的配方。类不仅要定义对象的接口（interface）（开发者访问的属性和方法），还要定义对象的内部工作（使属性和方法发挥作用的代码）。编译器和解释程序都根据类的说明构建对象。

实例

程序使用类创建对象时，生成的对象叫作类的实例（instance）。对类生成的对象的个数的唯一限制来自于运行代码的机器的物理内存。每个实例的行为相同，但实例处理一组独立的数据。由类创建对象实例的过程叫做实例化（instantiation）。

在前面的章节我们提到过，ECMAScript 并没有正式的类。相反，ECMA-262 把对象定义描述为对象的配方。这是 ECMAScript 逻辑上的一种折中方案，因为对象定义实际上是对象自身。即使类并不真正存在，我们也把对象定义叫做类，因为大多数开发者对此术语更熟悉，而且从功能上说，两者是等价的。

面向对象语言的要求

一种面向对象语言需要向开发者提供四种基本能力：

1. 封装 - 把相关的信息（无论数据或方法）存储在对象中的能力
2. 聚集 - 把一个对象存储在另一个对象内的能力
3. 继承 - 由另一个类（或多个类）得来类的属性和方法的能力
4. 多态 - 编写能以多种方法运行的函数或方法的能力

ECMAScript 支持这些要求，因此可被是看做面向对象的。

对象的构成

在 ECMAScript 中，对象由特性（attribute）构成，特性可以是原始值，也可以是引用值。如果特性存放的是函数，它将被看作对象的方法（method），否则该特性被看作对象的属性（property）。

ECMAScript 对象应用

对象的创建和销毁都在 **JavaScript** 执行过程中发生，理解这种范式的含义对理解整个语言至关重要。

声明和实例化

对象的创建方式是用关键字 new 后面跟上实例化的类的名字：

```
var oObject = new Object();
var oStringObject = new String();
```

第一行代码创建了 Object 类的一个实例，并把它存储到变量 oObject 中。第二行代码创建了 String 类的一个实例，把它存储在变量 oStringObject 中。如果构造函数无参数，括号则不是必需的。因此可以采用下面的形式重写上面的两行代码：

```
var oObject = new Object;
var oStringObject = new String;
```

对象引用

在前面的章节中，我们介绍了引用类型的概念。在 ECMAScript 中，不能访问对象的物理表示，只能访问对象的引用。每次创建对象，存储在变量中的都是该对象的引用，而不是对象本身。

对象废除

ECMAScript 拥有无用存储单元收集程序（garbage collection routine），意味着不必专门销毁对象来释放内存。当再没有对对象的引用时，称该对象被废除（dereference）了。运行无用存储单元收集程序时，所有废除的对象都被销毁。每当函数执行完它的代码，无用存储单元收集程序都会运行，释放所有的局部变量，还有在一些其他不可预知的情况下，无用存储单元收集程序也会运行。

把对象的所有引用都设置为 null，可以强制性地废除对象。例如：

```
var oObject = new Object;
// do something with the object here
oObject = null;
```

当变量 oObject 设置为 null 后，对第一个创建的对象的引用就不存在了。这意味着下次运行无用存储单元收集程序时，该对象将被销毁。

每用完一个对象后，就将其废除，来释放内存，这是个好习惯。这样还确保不再使用已经不能访问的对象，从而防止程序设计错误的出现。此外，旧的浏览器（如 IE/MAC）没有全面的无用存储单元收集程序，所以在卸载页面时，对象可能不能被正确销毁。废除对象和它的所有特性是确保内存使用正确的最好方法。

注意：废除对象的所有引用时要当心。如果一个对象有两个或更多引用，则要正确废除该对象，必须将其所有引用都设置为 null。

早绑定和晚绑定

所谓绑定（binding），即把对象的接口与对象实例结合在一起的方法。

早绑定（early binding）是指在实例化对象之前定义它的属性和方法，这样编译器或解释程序就能够提前转换机器代码。在 Java 和 Visual Basic 这样的语言中，有了早绑定，就可以在开发环境中使用 IntelliSense（即给开发者提供对象中属性和方法列表的功能）。ECMAScript 不是强类型语言，所以不支持早绑定。

另一方面，晚绑定（late binding）指的是编译器或解释程序在运行前，不知道对象的类型。使用晚绑定，无需检查对象的类型，只需检查对象是否支持属性和方法即可。ECMAScript 中的所有变量都采用晚绑定方法。这样就允许执行大量的对象操作，而无任何惩罚。

ECMAScript 对象类型

在 **ECMAScript** 中，所有对象并非同等创建的。

一般来说，可以创建并使用的对象有三种：本地对象、内置对象和宿主对象。

本地对象

ECMA-262 把本地对象（native object）定义为“独立于宿主环境的 ECMAScript 实现提供的对象”。简单来说，本地对象就是 ECMA-262 定义的类（引用类型）。它们包括：

- Object
- Function
- Array
- String
- Boolean
- Number
- Date
- RegExp
- Error
- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

相关页面

JavaScript 高级教程：ECMAScript 引用类型

JavaScript 高级教程：ECMAScript Function 类

JavaScript 参考手册：Array 对象

JavaScript 参考手册：Boolean 对象

JavaScript 参考手册：Date 对象

JavaScript 参考手册：Number 对象

JavaScript 参考手册：String 对象

JavaScript 参考手册：RegExp 对象

内置对象

ECMA-262 把内置对象（built-in object）定义为“由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现”。这意味着开发者不必明确实例化内置对象，它已被实例化了。ECMA-262 只定义了两个内置对象，即 Global 和 Math （它们也是本地对象，根据定义，每个内置对象都是本地对象）。

相关页面

JavaScript 参考手册：Global 对象

JavaScript 参考手册：Math 对象

宿主对象

所有非本地对象都是宿主对象（host object），即由 ECMAScript 实现的宿主环境提供的对象。

所有 BOM 和 DOM 对象都是宿主对象。

相关页面

JavaScript 高级教程：JavaScript 实现

W3School 参考手册：JavaScript 参考手册

W3School 教程：HTML DOM 教程

ECMAScript 对象作用域

作用域指的是变量的适用范围。

公用、私有和受保护作用域

概念

在传统的面向对象程序设计中，主要关注于公用和私有作用域。公用作用域中的对象属性可以从对象外部访问，即开发者创建对象的实例后，就可使用它的公用属性。而私有作用域中的属性只能在对象内部访问，即对于外部世界来说，这些属性并不存在。这意味着如果类定义了私有属性和方法，则它的子类也不能访问这些属性和方法。

受保护作用域也是用于定义私有的属性和方法，只是这些属性和方法还能被其子类访问。

ECMAScript 只有公用作用域

对 ECMAScript 讨论上面这些作用域几乎毫无意义，因为 ECMAScript 中只存在一种作用域 - 公用作用域。ECMAScript 中的所有对象的所有属性和方法都是公用的。因此，定义自己的类和对象时，必须格外小心。记住，所有属性和方法默认都是公用的！

建议性的解决方法

许多开发者都在网上提出了有效的属性作用域模式，解决了 ECMAScript 的这种问题。

由于缺少私有作用域，开发者确定了一个规约，说明哪些属性和方法应该被看做私有的。这种规约规定在属性前后加下划线：

```
obj._color_ = "blue";
```

这段代码中，属性 color 是私有的。注意，下划线并不改变属性是公用属性的事实，它只是告诉其他开发者，应该把该属性看作私有的。

有些开发者还喜欢用单下划线说明私有成员，例如：obj._color。

静态作用域

静态作用域定义的属性和方法任何时候都能从同一位置访问。在 Java 中，类可具有属性和方法，无需实例化该类的对象，即可访问这些属性和方法，例如 java.net.URLEncoder 类，它的函数 encode() 就是静态方法。

ECMAScript 没有静态作用域

严格来说，ECMAScript 并没有静态作用域。不过，它可以给构造函数提供属性和方法。还记得吗，构造函数只是函数。函数是对象，对象可以有属性和方法。例如：

```
function sayHello() {
  alert("hello");
}

sayHello.alternate = function() {
  alert("hi");
}

sayHello();         //输出 "hello"
sayHello.alternate();    //输出 "hi"
```

TIY

这里，方法 alternate() 实际上是函数 sayHello 的方法。可以像调用常规函数一样调用 sayHello() 输出 "hello"，也可以调用 sayHello.alternate() 输出 "hi"。即使如此，alternate() 也是 sayHello() 公用作用域中的方法，而不是静态方法。

关键字 this

this 的功能

在 ECMAScript 中，要掌握的最重要的概念之一是关键字 this 的用法，它用在对象的方法中。关键字 this 总是指向调用该方法的对象，例如：

```
var oCar = new Object;
oCar.color = "red";
oCar.showColor = function() {
  alert(`this.color`);
};

oCar.showColor();        //输出 "red"
```

TIY

在上面的代码中，关键字 this 用在对象的 showColor() 方法中。在此环境中，this 等于 oCar。下面的代码与上面的代码的功能相同：

```
var oCar = new Object;
oCar.color = "red";
oCar.showColor = function() {
  alert(`oCar.color`);
};

oCar.showColor();        //输出 "red"
```

TIY

使用 this 的原因

为什么使用 this 呢？因为在实例化对象时，总是不能确定开发者会使用什么样的变量名。使用 this，即可在任何多个地方重用同一个函数。请思考下面的例子：

```
function showColor() {
  alert(`this.color`);
};

var oCar1 = new Object;
oCar1.color = "red";
oCar1.showColor = showColor;

var oCar2 = new Object;
oCar2.color = "blue";
oCar2.showColor = showColor;

oCar1.showColor();        //输出 "red"
oCar2.showColor();        //输出 "blue"
```

TIY

在上面的代码中，首先用 this 定义函数 showColor()，然后创建两个对象（oCar1 和 oCar2），一个对象的 color 属性被设置为 "red"，另一个对象的 color 属性被设置为 "blue"。两个对象都被赋予了属性 showColor，指向原始的 showColor () 函数（注意这里不存在命名问题，因为一个是全局函数，而另一个是对象的属性）。调用每个对象的 showColor()，oCar1 输出是 "red"，而 oCar2 的输出是 "blue"。这是因为调用 oCar1.showColor() 时，函数中的 this 关键字等于 oCar1。调用 oCar2.showColor() 时，函数中的 this 关键字等于 oCar2。

注意，引用对象的属性时，必须使用 this 关键字。例如，如果采用下面的代码，showColor() 方法不能运行：

```
function showColor() {
  alert(`color`);
};
```

如果不用对象或 this 关键字引用变量，ECMAScript 就会把它看作局部变量或全局变量。然后该函数将查找名为 color 的局部或全局变量，但是不会找到。结果如何呢？该函数将在警告中显示 "null"。

ECMAScript 定义类或对象

使用预定义对象只是面向对象语言的能力的一部分，它真正强大之处在于能够创建自己专用的类和对象。

ECMAScript 拥有很多创建对象或类的方法。

工厂方式

原始的方式

因为对象的属性可以在对象创建后动态定义，所有许多开发者都在 JavaScript 最初引入时编写类似下面的代码：

```
var oCar = new Object;
oCar.color = "blue";
oCar.doors = 4;
oCar.mpg = 25;
oCar.showColor = function() {
  alert(this.color);
};
```

TIY

在上面的代码中，创建对象 car。然后给它设置几个属性：它的颜色是蓝色，有四个门，每加仑油可以跑 25 英里。最后一个属性实际上是指向函数的指针，意味着该属性是个方法。执行这段代码后，就可以使用对象 car。

不过这里有一个问题，就是可能需要创建多个 car 的实例。

解决方案：工厂方式

要解决该问题，开发者创造了能创建并返回特定类型的对象的工厂函数（factory function）。

例如，函数 createCar() 可用于封装前面列出的创建 car 对象的操作：

```
function createCar() {
  var oTempCar = new Object;
  oTempCar.color = "blue";
  oTempCar.doors = 4;
  oTempCar.mpg = 25;
  oTempCar.showColor = function() {
    alert(this.color);
  };
  return oTempCar;
}

var oCar1 = createCar();
var oCar2 = createCar();
```

TIY

在这里，第一个例子中的所有代码都包含在 createCar() 函数中。此外，还有一行额外的代码，返回 car 对象（oTempCar）作为函数值。调用此函数，将创建新对象，并赋予它所有必要的属性，复制出一个我们在前面说明过的 car 对象。因此，通过这种方法，我们可以很容易地创建 car 对象的两个版本（oCar1 和 oCar2），它们的属性完全一样。

为函数传递参数

我们还可以修改 createCar() 函数，给它传递各个属性的默认值，而不是简单地赋予属性默认值：

```
function createCar(sColor,iDoors,iMpg) {
  var oTempCar = new Object;
  oTempCar.color = sColor;
  oTempCar.doors = iDoors;
  oTempCar.mpg = iMpg;
  oTempCar.showColor = function() {
    alert(this.color);
  };
  return oTempCar;
}

var oCar1 = createCar("red",4,23);
var oCar2 = createCar("blue",3,25);

oCar1.showColor();        //输出 "red"
oCar2.showColor();        //输出 "blue"
```

TIY

给 createCar() 函数加上参数，即可为要创建的 car 对象的 color、doors 和 mpg 属性赋值。这使两个对象具有相同的属性，却有不同的属性值。

在工厂函数外定义对象的方法

虽然 ECMAScript 越来越正式化，但创建对象的方法却被置之不理，且其规范化至今还遭人反对。一部分是语义上的原因（它看起来不像使用带有构造函数 new 运算符那么正规），一部分是功能上的原因。功能原因在于用这种方式必须创建对象的方法。前面的例子中，每次调用函数 createCar()，都要创建新函数 showColor()，意味着每个对象都有自己的 showColor() 版本。而事实上，每个对象都共享同一个函数。

有些开发者在工厂函数外定义对象的方法，然后通过属性指向该方法，从而避免这个问题：

```
`function showColor() {
  alert(this.color);
}`

function createCar(sColor,iDoors,iMpg) {
  var oTempCar = new Object;
  oTempCar.color = sColor;
  oTempCar.doors = iDoors;
  oTempCar.mpg = iMpg;
  `oTempCar.showColor = showColor;`
  return oTempCar;
}

var oCar1 = createCar("red",4,23);
var oCar2 = createCar("blue",3,25);

oCar1.showColor();        //输出 "red"
oCar2.showColor();        //输出 "blue"
```

TIY

在上面这段重写的代码中，在函数 createCar() 之前定义了函数 showColor()。在 createCar() 内部，赋予对象一个指向已经存在的 showColor() 函数的指针。从功能上讲，这样解决了重复创建函数对象的问题；但是从语义上讲，该函数不太像是对象的方法。

所有这些问题都引发了开发者定义的构造函数的出现。

构造函数方式

创建构造函数就像创建工厂函数一样容易。第一步选择类名，即构造函数的名字。根据惯例，这个名字的首字母大写，以使它与首字母通常是小写的变量名分开。除了这点不同，构造函数看起来很像工厂函数。请考虑下面的例子：

```
function Car(sColor,iDoors,iMpg) {
  `this`.color = sColor;
  `this`.doors = iDoors;
  `this`.mpg = iMpg;
  `this`.showColor = function() {
    alert(this.color);
  };
}

var oCar1 = `new` Car("red",4,23);
var oCar2 = `new` Car("blue",3,25);
```

TIY

下面为您解释上面的代码与工厂方式的差别。首先在构造函数内没有创建对象，而是使用 this 关键字。使用 new 运算符构造函数时，在执行第一行代码前先创建一个对象，只有用 this 才能访问该对象。然后可以直接赋予 this 属性，默认情况下是构造函数的返回值（不必明确使用 return 运算符）。

现在，用 new 运算符和类名 Car 创建对象，就更像 ECMAScript 中一般对象的创建方式了。

你也许会问，这种方式在管理函数方面是否存在于前一种方式相同的问题呢？是的。

就像工厂函数，构造函数会重复生成函数，为每个对象都创建独立的函数版本。不过，与工厂函数相似，也可以用外部函数重写构造函数，同样地，这么做语义上无任何意义。这正是下面要讲的原型方式的优势所在。

原型方式

该方式利用了对象的 prototype 属性，可以把它看成创建新对象所依赖的原型。

这里，首先用空构造函数来设置类名。然后所有的属性和方法都被直接赋予 prototype 属性。我们重写了前面的例子，代码如下：

```
function Car() {
}

Car.prototype.color = "blue";
Car.prototype.doors = 4;
Car.prototype.mpg = 25;
Car.prototype.showColor = function() {
  alert(this.color);
};

var oCar1 = new Car();
var oCar2 = new Car();
```

TIY

在这段代码中，首先定义构造函数（Car），其中无任何代码。接下来的几行代码，通过给 Car 的 prototype 属性添加属性去定义 Car 对象的属性。调用 new Car() 时，原型的所有属性都被立即赋予要创建的对象，意味着所有 Car 实例存放的都是指向 showColor() 函数的指针。从语义上讲，所有属性看起来都属于一个对象，因此解决了前面两种方式存在的问题。

此外，使用这种方式，还能用 instanceof 运算符检查给定变量指向的对象的类型。因此，下面的代码将输出 TRUE：

```
alert(oCar1 instanceof Car);    //输出 "true"
```

原型方式的问题

原型方式看起来是个不错的解决方案。遗憾的是，它并不尽如人意。

首先，这个构造函数没有参数。使用原型方式，不能通过给构造函数传递参数来初始化属性的值，因为 Car1 和 Car2 的 color 属性都等于 "blue"，doors 属性都等于 4，mpg 属性都等于 25。这意味着必须在对象创建后才能改变属性的默认值，这点很令人讨厌，但还没完。真正的问题出现在属性指向的是对象，而不是函数时。函数共享不会造成问题，但对象却很少被多个实例共享。请思考下面的例子：

```
function Car() {
}

Car.prototype.color = "blue";
Car.prototype.doors = 4;
Car.prototype.mpg = 25;
`Car.prototype.drivers = new Array("Mike","John");`
Car.prototype.showColor = function() {
  alert(this.color);
};

var oCar1 = new Car();
var oCar2 = new Car();

`oCar1.drivers.push("Bill");`

alert(oCar1.drivers);    //输出 "Mike,John,Bill"
alert(oCar2.drivers);    //输出 "Mike,John,Bill"
```

TIY

上面的代码中，属性 drivers 是指向 Array 对象的指针，该数组中包含两个名字 "Mike" 和 "John"。由于 drivers 是引用值，Car 的两个实例都指向同一个数组。这意味着给 oCar1.drivers 添加值 "Bill"，在 oCar2.drivers 中也能看到。输出这两个指针中的任何一个，结果都是显示字符串 "Mike,John,Bill"。

由于创建对象时有这么多问题，你一定会想，是否有种合理的创建对象的方法呢？答案是有，需要联合使用构造函数和原型方式。

混合的构造函数/原型方式

联合使用构造函数和原型方式，就可像用其他程序设计语言一样创建对象。这种概念非常简单，即用构造函数定义对象的所有非函数属性，用原型方式定义对象的函数属性（方法）。结果是，所有函数都只创建一次，而每个对象都具有自己的对象属性实例。

我们重写了前面的例子，代码如下：

```
function Car(sColor,iDoors,iMpg) {
  this.color = sColor;
  this.doors = iDoors;
  this.mpg = iMpg;
  this.drivers = new Array("Mike","John");
}

Car.prototype.showColor = function() {
  alert(this.color);
};

var oCar1 = new Car("red",4,23);
var oCar2 = new Car("blue",3,25);

oCar1.drivers.push("Bill");

alert(oCar1.drivers);    //输出 "Mike,John,Bill"
alert(oCar2.drivers);    //输出 "Mike,John"
```

TIY

现在就更像创建一般对象了。所有的非函数属性都在构造函数中创建，意味着又能够用构造函数的参数赋予属性默认值了。因为只创建 showColor() 函数的一个实例，所以没有内存浪费。此外，给 oCar1 的 drivers 数组添加 "Bill" 值，不会影响到 oCar2 的数组，所以输出这些数组的值时，oCar1.drivers 显示的是 "Mike,John,Bill"，而 oCar2.drivers 显示的是 "Mike,John"。因为使用了原型方式，所以仍然能利用 instanceof 运算符来判断对象的类型。

这种方式是 ECMAScript 采用的主要方式，它具有其他方式的特性，却没有他们的副作用。不过，有些开发者仍觉得这种方法不够完美。

动态原型方法

对于习惯使用其他语言的开发者来说，使用混合的构造函数/原型方式感觉不那么和谐。毕竟，定义类时，大多数面向对象语言都对属性和方法进行了视觉上的封装。请考虑下面的 Java 类：

```
class Car {
  public String color = "blue";
  public int doors = 4;
  public int mpg = 25;

  public Car(String color, int doors, int mpg) {
    this.color = color;
    this.doors = doors;
    this.mpg = mpg;
  }

  public void showColor() {
    System.out.println(color);
  }
}
```

Java 很好地打包了 Car 类的所有属性和方法，因此看见这段代码就知道它要实现什么功能，它定义了一个对象的信息。批评混合的构造函数/原型方式的人认为，在构造函数内部找属性，在其外部找方法的做法不合逻辑。因此，他们设计了动态原型方法，以提供更友好的编码风格。

动态原型方法的基本想法与混合的构造函数/原型方式相同，即在构造函数内定义非函数属性，而函数属性则利用原型属性定义。唯一的区别是赋予对象方法的位置。下面是用动态原型方法重写的 Car 类：

```
function Car(sColor,iDoors,iMpg) {
  this.color = sColor;
  this.doors = iDoors;
  this.mpg = iMpg;
  this.drivers = new Array("Mike","John");

  if (`typeof Car._initialized == "undefined"`) {
    Car.prototype.showColor = function() {
      alert(this.color);
    };

    `Car._initialized = true;`
  }
}
```

TIY

直到检查 typeof Car._initialized 是否等于 "undefined" 之前，这个构造函数都未发生变化。这行代码是动态原型方法中最重要的部分。如果这个值未定义，构造函数将用原型方式继续定义对象的方法，然后把 Car._initialized 设置为 true。如果这个

值定义了（它的值为 true 时，typeof 的值为 Boolean），那么就不再创建该方法。简而言之，该方法使用标志（_initialized）来判断是否已给原型赋予了任何方法。该方法只创建并赋值一次，传统的 OOP 开发者会高兴地发现，这段代码看起来更像其他语言中的类定义了。

混合工厂方式

这种方式通常是在不能应用前一种方式时的变通方法。它的目的是创建假构造函数，只返回另一种对象的新实例。

这段代码看起来与工厂函数非常相似：

```
function Car() {
  `var oTempCar = new Object;`
  oTempCar.color = "blue";
  oTempCar.doors = 4;
  oTempCar.mpg = 25;
  oTempCar.showColor = function() {
    alert(this.color);
  };

  return oTempCar;
}
```

TIY

与经典方式不同，这种方式使用 new 运算符，使它看起来像真正的构造函数：

```
var car = new Car();
```

由于在 Car() 构造函数内部调用了 new 运算符，所以将忽略第二个 new 运算符（位于构造函数之外），在构造函数内部创建的对象被传递回变量 car。

这种方式在对象方法的内部管理方面与经典方式有着相同的问题。强烈建议：除非万不得已，还是避免使用这种方式。

采用哪种方式

如前所述，目前使用最广泛的是混合的构造函数/原型方式。此外，动态原始方法也很流行，在功能上与构造函数/原型方式等价。可以采用这两种方式中的任何一种。不过不要单独使用经典的构造函数或原型方式，因为这样会给代码引入问题。

实例

对象令人感兴趣的一点是用它们解决问题的方式。ECMAScript 中最常见的一个问题是字符串连接的性能。与其他语言类似，ECMAScript 的字符串是不可变的，即它们的值不能改变。请考虑下面的代码：

```
var str = "hello ";
str += "world";
```

实际上，这段代码在幕后执行的步骤如下：

1. 创建存储 "hello " 的字符串。
2. 创建存储 "world" 的字符串。
3. 创建存储连接结果的字符串。
4. 把 str 的当前内容复制到结果中。
5. 把 "world" 复制到结果中。
6. 更新 str，使它指向结果。

每次完成字符串连接都会执行步骤 2 到 6，使得这种操作非常消耗资源。如果重复这一过程几百次，甚至几千次，就会造成性能问题。解决方法是用 Array 对象存储字符串，然后用 join() 方法（参数是空字符串）创建最后的字符串。想象用下面的代码代替前面的代码：

```
var arr = new Array();
arr[0] = "hello ";
arr[1] = "world";
var str = arr.join("");
```

这样，无论数组中引入多少字符串都不成问题，因为只在调用 join() 方法时才会发生连接操作。此时，执行的步骤如下：

1. 创建存储结果的字符串
2. 把每个字符串复制到结果中的合适位置

虽然这种解决方案很好，但还有更好的方法。问题是，这段代码不能确切反映出它的意图。要使它更容易理解，可以用 StringBuffer 类打包该功能：

```
function StringBuffer () {
  this._strings_ = new Array();
}

StringBuffer.prototype.append = function(str) {
  this._strings_.push(str);
};

StringBuffer.prototype.toString = function() {
  return this._strings_.join("");
};
```

这段代码首先要注意的是 strings 属性，本意是私有属性。它只有两个方法，即 append() 和 toString() 方法。append() 方法有一个参数，它把该参数附加到字符串数组中，toString() 方法调用数组的 join 方法，返回真正连接成的字符串。要用 StringBuffer 对象连接一组字符串，可以用下面的代码：

```
var buffer = new StringBuffer ();
buffer.append("hello ");
buffer.append("world");
var result = buffer.toString();
```

TIY

可用下面的代码测试 StringBuffer 对象和传统的字符串连接方法的性能：

```
var d1 = new Date();
var str = "";
for (var i=0; i < 10000; i++) {
    str += "text";
}
var d2 = new Date();

document.write("Concatenation with plus: "
 + (d2.getTime() - d1.getTime()) + " milliseconds");

var buffer = new StringBuffer();
d1 = new Date();
for (var i=0; i < 10000; i++) {
    buffer.append("text");
}
var result = buffer.toString();
d2 = new Date();

document.write("<br />Concatenation with StringBuffer: "
 + (d2.getTime() - d1.getTime()) + " milliseconds");
```

TIY

这段代码对字符串连接进行两个测试，第一个使用加号，第二个使用 StringBuffer 类。每个操作都连接 10000 个字符串。日期值 d1 和 d2 用于判断完成操作需要的时间。请注意，创建 Date 对象时，如果没有参数，赋予对象的是当前的日期和时间。要计算连接操作历经多少时间，把日期的毫秒表示（用 getTime() 方法的返回值）相减即可。这是衡量 JavaScript 性能的常见方法。该测试的结果可以帮助您比较使用 StringBuffer 类与使用加号的效率差异。

ECMAScript 修改对象

通过使用 **ECMAScript**，不仅可以创建对象，还可以修改已有对象的行为。

prototype 属性不仅可以定义构造函数的属性和方法，还可以为本地对象添加属性和方法。

创建新方法

通过已有的方法创建新方法

可以用 prototype 属性为任何已有的类定义新方法，就像处理自己的类一样。例如，还记得 Number 类的 toString() 方法吗？如果给它传递参数 16，它将输出十六进制的字符串。如果这个方法的参数是 2，那么它将输出二进制的字符串。我们可以创建一个方法，可以把数字对象直接转换为十六进制字符串。创建这个方法非常简单：

```
Number.prototype.toHexString = function() {
  return this.toString(16);
};
```

在此环境中，关键字 this 指向 Number 的实例，因此可完全访问 Number 的所有方法。有了这段代码，可实现下面的操作：

```
var iNum = 15;
alert(iNum.toHexString());        //输出 "F"
```

TIY

由于数字 15 等于十六进制中的 F，因此警告将显示 "F"。

重命名已有方法

我们还可以为已有的方法命名更易懂的名称。例如，可以给 Array 类添加两个方法 enqueue() 和 dequeue()，只让它们反复调用已有的 push() 和 shift() 方法即可：

```
Array.prototype.enqueue = function(vItem) {
  this.push(vItem);
};

Array.prototype.dequeue = function() {
  return this.shift();
};
```

TIY

添加与已有方法无关的方法

当然，还可以添加与已有方法无关的方法。例如，假设要判断某个项在数组中的位置，没有本地方法可以做这种事情。我们可以轻松地创建下面的方法：

```
Array.prototype.indexOf = function (vItem) {
  for (var i=0; i<this.length; i++) {
    if (vItem == this[i]) {
      return i;
    }
  }

  return -1;
}
```

该方法 indexOf() 与 String 类的同名方法保持一致，在数组中检索每个项，直到发现与传进来的项相同的项目为止。如果找到相同的项，则返回该项的位置，否则，返回 -1。有了这种定义，我们可以编写下面的代码：

```
var aColors = new Array("red","green","blue");
alert(aColors.indexOf("green"));    //输出 "1"
```

TIY

为本地对象添加新方法

最后，如果想给 ECMAScript 中每个本地对象添加新方法，必须在 Object 对象的 prototype 属性上定义它。前面的章节我们讲过，所有本地对象都继承了 Object 对象，所以对 Object 对象做任何改变，都会反应在所有本地对象上。例如，如果想添加一个用警告输出对象的当前值的方法，可以采用下面的代码：

```
Object.prototype.showValue = function () {
  alert(this.valueOf());
};

var str = "hello";
var iNum = 25;
str.showValue();        //输出 "hello"
iNum.showValue();        //输出 "25"
```

TIY

这里，String 和 Number 对象都从 Object 对象继承了 showValue() 方法，分别在它们的对象上调用该方法，将显示 "hello" 和 "25"。

重定义已有方法

就像能给已有的类定义新方法一样，也可重定义已有的方法。如前面的章节所述，函数名只是指向函数的指针，因此可以轻松地指向其他函数。如果修改了本地方法，如 toString()，会出现什么情况呢？

```
Function.prototype.toString = function() {
  return "Function code hidden";
}
```

前面的代码完全合法，运行结果完全符合预期：

```
function sayHi() {
  alert("hi");
}

alert(sayHi.toString());    //输出 "Function code hidden"
```

TIY

也许你还记得，Function 对象这一章中介绍过 Function 的 toString() 方法通常输出的是函数的源代码。覆盖该方法，可以返回另一个字符串（在这个例子中，可以返回 "Function code hidden"）。不过，toString() 指向的原始函数怎么了呢？它将被无用存储单元回收程序回收，因为它被完全废弃了。没有能够恢复原始函数的方法，所以在覆盖原始方法前，比较安全的做法是存储它的指针，以便以后的使用。有时你甚至可能在新方法中调用原始方法：

```
`Function.prototype.originalToString = Function.prototype.toString;

Function.prototype.toString = function() {
  if (this.originalToString().length > 100) {
    return "Function too long to display.";
  } else {
    return this.originalToString();
  }
};
```

TIY

在这段代码中，第一行代码把对当前 toString() 方法的引用保存在属性 originalToString 中。然后用定制的方法覆盖了 toString() 方法。新方法将检查该函数源代码的长度是否大于 100。如果是，就返回错误信息，说明该函数代码太长，否则调用 originalToString() 方法，返回函数的源代码。

极晚绑定（Very Late Binding）

从技术上讲，根本不存在极晚绑定。本书采用该术语描述 ECMAScript 中的一种现象，即能够在对象实例化后再定义它的方法。例如：

```
var o = new Object();

Object.prototype.sayHi = function () {
  alert("hi");
};

o.sayHi();
```

TIY

在大多数程序设计语言中，必须在实例化对象之前定义对象的方法。这里，方法 sayHi() 是在创建 Object 类的一个实例之后来添加进来的。在传统语言中不仅没听说过这种操作，也没听说过该方法还会自动赋予 Object 对象的实例并能立即使用（接下来的一行）。

注意：不建议使用极晚绑定方法，因为很难对其跟踪和记录。不过，还是应该了解这种可能。

ECMAScript 继承

ECMAScript 继承机制实例

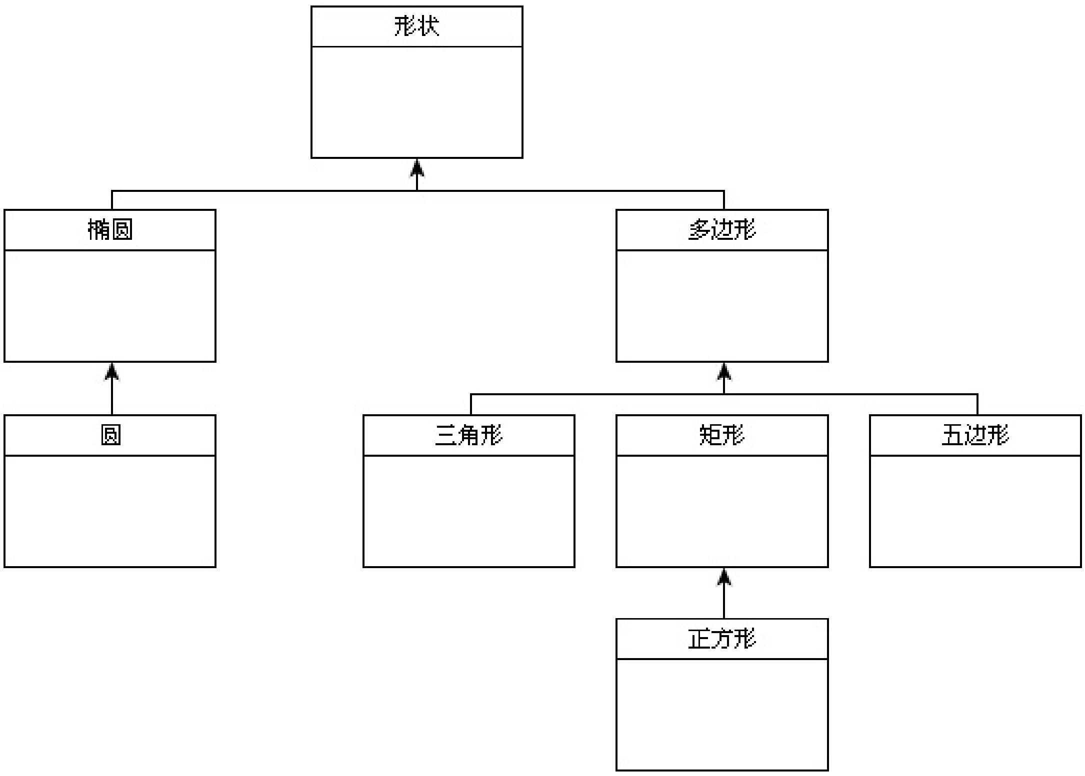
本节使用一个经典的例子解释 **ECMAScript** 的继承机制。

继承机制实例

说明继承机制最简单的方式是，利用一个经典的例子 - 几何形状。实际上，几何形状只有两种，即椭圆形（是圆形的）和多边形（具有一定数量的边）。圆是椭圆的一种，它只有一个焦点。三角形、矩形和五边形都是多边形的一种，具有不同数量的边。正方形是矩形的一种，所有的边等长。这就构成了一种完美的继承关系。

在这个例子中，形状（Shape）是椭圆形（Ellipse）和多边形（Polygon）的基类（base class）（所有类都由它继承而来）。椭圆具有一个属性 *foci*，说明椭圆具有的焦点的个数。圆形（Circle）继承了椭圆形，因此圆形是椭圆形的子类（subclass），椭圆形是圆形的超类（superclass）。同样，三角形（Triangle）、矩形（Rectangle）和五边形（Pentagon）都是多边形的子类，多边形是它们的超类。最后，正方形（Square）继承了矩形。

最好用图来解释这种继承关系，这是 UML（统一建模语言）的用武之地。UML 的主要用途之一是，可视化地表示像继承这样的复杂对象关系。下面的图示是解释 Shape 和它的子类之间关系的 UML 图示：



在 UML 中，每个方框表示一个类，由类名说明。三角形 、矩形和五边形顶部的线段汇集在一起，指向形状，说明这些类都由形状继承而来。同样，从正方形指向矩形的箭头说明了它们之间的继承关系。

ECMAScript 继承机制实现

继承机制的实现

要用 ECMAScript 实现继承机制，您可以从要继承的基类入手。所有开发者定义的类都可作为基类。出于安全原因，本地类和宿主类不能作为基类，这样可以防止公用访问编译过的浏览器级的代码，因为这些代码可以被用于恶意攻击。

选定基类后，就可以创建它的子类了。是否使用基类完全由你决定。有时，你可能想创建一个不能直接使用的基类，它只是用于给子类提供通用的函数。在这种情况下，基类被看作抽象类。

尽管 ECMAScript 并没有像其他语言那样严格地定义抽象类，但有时它的确会创建一些不允许使用的类。通常，我们称这种类为抽象类。

创建的子类将继承超类的所有属性和方法，包括构造函数及方法的实现。记住，所有属性和方法都是公用的，因此子类可直接访问这些方法。子类还可添加超类中没有的新属性和方法，也可以覆盖超类的属性和方法。

继承的方式

和其他功能一样，ECMAScript 实现继承的方式不止一种。这是因为 JavaScript 中的继承机制并不是明确规定的，而是通过模仿实现的。这意味着所有的继承细节并非完全由解释程序处理。作为开发者，你有权决定最适用的继承方式。

下面为您介绍几种具体的继承方式。

对象冒充

构想原始的 ECMAScript 时，根本没打算设计对象冒充（object masquerading）。它是在开发者开始理解函数的工作方式，尤其是如何在函数环境中使用 this 关键字后才发展出来。

其原理如下：构造函数使用 this 关键字给所有属性和方法赋值（即采用类声明的构造函数方式）。因为构造函数只是一个函数，所以可使 ClassA 构造函数成为 ClassB 的方法，然后调用它。ClassB 就会收到 ClassA 的构造函数中定义的属性和方法。例如，用下面的方式定义 ClassA 和 ClassB：

```
function ClassA(sColor) {
    this.color = sColor;
    this.sayColor = function () {
        alert(this.color);
    };
}

function ClassB(sColor) {
}
```

还记得吗？关键字 this 引用的是构造函数当前创建的对象。不过在这个方法中，this 指向的所属的对象。这个原理是把 ClassA 作为常规函数来建立继承机制，而不是作为构造函数。如下使用构造函数 ClassB 可以实现继承机制：

```
function ClassB(sColor) {
    this.newMethod = ClassA;
    this.newMethod(sColor);
    delete this.newMethod;
}
```

在这段代码中，为 ClassA 赋予了方法 newMethod（请记住，函数名只是指向它的指针）。然后调用该方法，传递给它的是 ClassB 构造函数的参数 sColor。最后一行代码删除了对 ClassA 的引用，这样以后就不能再调用它。

所有新属性和新方法都必须在删除了新方法的代码行后定义。否则，可能会覆盖超类的相关属性和方法：

```
function ClassB(sColor, sName) {
    this.newMethod = ClassA;
    this.newMethod(sColor);
    delete this.newMethod;

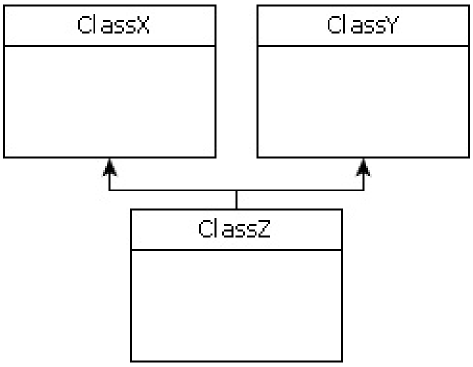
    this.name = sName;
    this.sayName = function () {
        alert(this.name);
    };
}
```

为证明前面的代码有效，可以运行下面的例子：

```
var objA = new ClassA("blue");
var objB = new ClassB("red", "John");
objA.sayColor();    //输出 "blue"
objB.sayColor();    //输出 "red"
objB.sayName();        //输出 "John"
```

对象冒充可以实现多重继承

有趣的是，对象冒充可以支持多重继承。也就是说，一个类可以继承多个超类。用 UML 表示的多重继承机制如下图所示：



例如，如果存在两个类 ClassX 和 ClassY，ClassZ 想继承这两个类，可以使用下面的代码：

```
function ClassZ() {
    this.newMethod = ClassX;
    this.newMethod();
    delete this.newMethod;

    this.newMethod = ClassY;
    this.newMethod();
    delete this.newMethod;
}
```

TIY

这里存在一个弊端，如果存在两个类 ClassX 和 ClassY 具有同名的属性或方法，ClassY 具有高优先级。因为它从后面的类继承。除这点小问题之外，用对象冒充实现多重继承机制轻而易举。

由于这种继承方法的流行，ECMAScript 的第三版为 Function 对象加入了两个方法，即 call() 和 apply()。

call() 方法

call() 方法是与经典的对象冒充方法最相似的方法。它的第一个参数用作 this 的对象。其他参数都直接传递给函数自身。例如：

```
function sayColor(sPrefix,sSuffix) {
    alert(sPrefix + this.color + sSuffix);
};

var obj = new Object();
obj.color = "blue";

sayColor.call(obj, "The color is ", "a very nice color indeed.");
```

在这个例子中，函数 sayColor() 在对象外定义，即使它不属于任何对象，也可以引用关键字 this。对象 obj 的 color 属性等于 blue。调用 call() 方法时，第一个参数是 obj，说明应该赋予 sayColor() 函数中的 this 关键字值是 obj。第二个和第三个参数是字符串。它们与 sayColor() 函数中的参数 sPrefix 和 sSuffix 匹配，最后生成的消息 "The color is blue, a very nice color indeed." 将被显示出来。

要与继承机制的对象冒充方法一起使用该方法，只需将前三行的赋值、调用和删除代码替换即可：

```
function ClassB(sColor, sName) {
    //this.newMethod = ClassA;
    //this.newMethod(color);
    //delete this.newMethod;
    ClassA.call(this, sColor);

    this.name = sName;
    this.sayName = function () {
        alert(this.name);
    };
}
```

TIY

这里，我们需要让 ClassA 中的关键字 this 等于新创建的 ClassB 对象，因此 this 是第一个参数。第二个参数 sColor 对两个类来说都是唯一的参数。

apply() 方法

apply() 方法有两个参数，用作 this 的对象和要传递给函数的参数的数组。例如：

```
function sayColor(sPrefix,sSuffix) {
    alert(sPrefix + this.color + sSuffix);
};

var obj = new Object();
obj.color = "blue";

sayColor.apply(obj, new Array("The color is ", "a very nice color i
```

这个例子与前面的例子相同，只是现在调用的是 apply() 方法。调用 apply() 方法时，第一个参数仍是 obj，说明应该赋予 sayColor() 函数中的 this 关键字值是 obj。第二个参数是由两个字符串构成的数组，与 sayColor() 函数中的参数 sPrefix 和 sSuffix 匹配，最后生成的消息仍是 "The color is blue, a very nice color indeed."，将被显示出来。

该方法也用于替换前三行的赋值、调用和删除新方法的代码：

```
function ClassB(sColor, sName) {
    //this.newMethod = ClassA;
    //this.newMethod(color);
    //delete this.newMethod;
    ClassA.apply(this, new Array(sColor));

    this.name = sName;
    this.sayName = function () {
        alert(this.name);
    };
}
```

同样的，第一个参数仍是 this，第二个参数是只有一个值 color 的数组。可以把 ClassB 的整个 arguments 对象作为第二个参数传递给 apply() 方法：

```
function ClassB(sColor, sName) {
    //this.newMethod = ClassA;
    //this.newMethod(color);
    //delete this.newMethod;
    ClassA.apply(this, arguments);

    this.name = sName;
    this.sayName = function () {
        alert(this.name);
    };
}
```

TIY

当然，只有超类中的参数顺序与子类中的参数顺序完全一致时才可以传递参数对象。如果不是，就必须创建一个单独的数组，按照正确的顺序放置参数。此外，还可使用 call() 方法。

原型链（prototype chaining）

继承这种形式在 ECMAScript 中原本是用于原型链的。上一章介绍了定义类的原型方式。原型链扩展了这种方式，以一种有趣的方式实现继承机制。

在上一章学过，prototype 对象是个模板，要实例化的对象都以这个模板为基础。总而言之，prototype 对象的任何属性和方法都被传递给那个类的所有实例。原型链利用这种功能来实现继承机制。

如果用原型方式重定义前面例子中的类，它们将变为下列形式：

```
function ClassA() {
}

ClassA.prototype.color = "blue";
ClassA.prototype.sayColor = function () {
    alert(this.color);
};

function ClassB() {
}

`ClassB.prototype = new ClassA();`
```

原型方式的神奇之处在于突出显示的蓝色代码行。这里，把 ClassB 的 prototype 属性设置成 ClassA 的实例。这很有意思，因为想要 ClassA 的所有属性和方法，但又不想逐个将它们 ClassB 的 prototype 属性。还有比把 ClassA 的实例赋予 prototype 属性更好的方法吗？

注意：调用 ClassA 的构造函数，没有给它传递参数。这在原型链中是标准做法。要确保构造函数没有任何参数。

与对象冒充相似，子类的所有属性和方法都必须出现在 prototype 属性被赋值后，因为在它之前赋值的所有方法都会被删除。为什么？因为 prototype 属性被替换成了新对象，添加了新方法的原始对象将被销毁。所以，为 ClassB 类添加 name 属性和 sayName() 方法的代码如下：

```
function ClassB() {
}

ClassB.prototype = new ClassA();

ClassB.prototype.name = "";
ClassB.prototype.sayName = function () {
    alert(this.name);
};
```

可通过运行下面的例子测试这段代码：

```
var objA = new ClassA();
var objB = new ClassB();
objA.color = "blue";
objB.color = "red";
objB.name = "John";
objA.sayColor();
objB.sayColor();
objB.sayName();
```

TIY

此外，在原型链中，instanceof 运算符的运行方式也很独特。对 ClassB 的所有实例，instanceof 为 ClassA 和 ClassB 都返回 true。例如：

```
var objB = new ClassB();
alert(objB instanceof ClassA);    //输出 "true"
alert(objB instanceof ClassB);    //输出 "true"
```

在 ECMAScript 的弱类型世界中，这是极其有用的工具，不过使用对象冒充时不能使用它。

原型链的弊端是不支持多重继承。记住，原型链会用另一类型的对象重写类的 prototype 属性。

混合方式

这种继承方式使用构造函数定义类，并非使用任何原型。对象冒充的主要问题是必须使用构造函数方式，这不是最好的选择。不过如果使用原型链，就无法使用带参数的构造函数了。开发者如何选择呢？答案很简单，两者都用。

在前一章，我们曾经讲解过创建类的最好方式是用构造函数定义属性，用原型定义方法。这种方式同样适用于继承机制，用对象冒充继承构造函数的属性，用原型链继承 prototype 对象的方法。用这两种方式重写前面的例子，代码如下：

```
function ClassA(sColor) {
    this.color = sColor;
}

ClassA.prototype.sayColor = function () {
    alert(this.color);
};

function ClassB(sColor, sName) {
    `ClassA.call(this, sColor);`
    this.name = sName;
}

`ClassB.prototype = new ClassA();`

ClassB.prototype.sayName = function () {
    alert(this.name);
};
```

在此例子中，继承机制由两行突出显示的蓝色代码实现。在第一行突出显示的代码中，在 ClassB 构造函数中，用对象冒充继承 ClassA 类的 sColor 属性。在第二行突出显示的代码中，用原型链继承 ClassA 类的方法。由于这种混合方式使用了原型链，所以 instanceof 运算符仍能正确运行。

下面的例子测试了这段代码：

```
var objA = new ClassA("blue");
var objB = new ClassB("red", "John");
objA.sayColor();    //输出 "blue"
objB.sayColor();    //输出 "red"
objB.sayName();    //输出 "John"
```

TIY

W3School jQuery 教程

来源：jQuery教程

整理：飞龙

jQuery 基础

jQuery 简介

jQuery 库可以通过一行简单的标记被添加到网页中。

jQuery 库 - 特性

jQuery 是一个 JavaScript 函数库。

jQuery 库包含以下特性：

- HTML 元素选取
- HTML 元素操作
- CSS 操作
- HTML 事件函数
- JavaScript 特效和动画
- HTML DOM 遍历和修改
- AJAX
- Utilities

向您的页面添加 jQuery 库

jQuery 库位于一个 JavaScript 文件中，其中包含了所有的 jQuery 函数。

可以通过下面的标记把 jQuery 添加到网页中：

```
<head>
`<script type="text/javascript" src="jquery.js"></script>`
</head>
```

请注意，<script> 标签应该位于页面的 <head> 部分。

基础 jQuery 实例

下面的例子演示了 jQuery 的 hide() 函数，隐藏了 HTML 文档中所有的 <p> 元素。

实例

```
<html>
<head>
`<script type="text/javascript" src="jquery.js"></script>`
<script type="text/javascript">
$(document).ready(function(){
$("button").click(function(){
`$("p").hide();`
});
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button type="button">Click me</button>
</body>
</html>
```

下载 jQuery

共有两个版本的 jQuery 可供下载：一份是精简过的，另一份是未压缩的（供调试或阅读）。

这两个版本都可从 jQuery.com 下载。

库的替代

Google 和 Microsoft 对 jQuery 的支持都很好。

如果您不愿意在自己的计算机上存放 jQuery 库，那么可以从 Google 或 Microsoft 加载 CDN jQuery 核心文件。

使用 Google 的 CDN

```
<head>
<script type="text/javascript" src="http://ajax.googleapis.com/aja
/jquery/1.4.0/jquery.min.js"></script>
</head>
```

使用 Microsoft 的 CDN

```
<head>
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/
/jquery-1.4.min.js"></script>
</head>
```

jQuery 安装

把 jQuery 添加到您的网页

如需使用 jQuery，您需要下载 jQuery 库（会在下面为您讲解），然后把它包含在希望使用的网页中。

jQuery 库是一个 JavaScript 文件，您可以使用 HTML 的 <script> 标签引用它：

```
<head>
<script src="jquery.js"></script>
</head>
```

请注意，<script> 标签应该位于页面的 <head> 部分。

提示：您是否很疑惑为什么我们没有在 <script> 标签中使用 type="text/javascript" ？

在 HTML5 中，不必那样做了。JavaScript 是 HTML5 以及所有现代浏览器中的默认脚本语言！

下载 jQuery

有两个版本的 jQuery 可供下载：

- Production version - 用于实际的网站中，已被精简和压缩。
- Development version - 用于测试和开发（未压缩，是可读的代码）

这两个版本都可以从 jQuery.com 下载。

提示：您可以把下载文件放到与页面相同的目录中，这样更方便使用。

替代方案

如果您不希望下载并存放 jQuery，那么也可以通过 CDN（内容分发网络） 引用它。

谷歌和微软的服务器都存有 jQuery 。

如需从谷歌或微软引用 jQuery，请使用以下代码之一：

Google CDN:

```
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.0/jque
</script>
</head>
```

提示：通过 Google CDN 来获得最新可用的版本：

如果您观察上什么的 Google URL - 在 URL 中规定了 jQuery 版本 (1.8.0)。如果您希望使用最新版本的 jQuery，也可以从版本字符串的末尾（比如本例 1.8）删除一个数字，谷歌会返回 1.8 系列中最新的可用版本（1.8.0、1.8.1 等等），或者也可以只剩第一个数字，那么谷歌会返回 1 系列中最新的可用版本（从 1.1.0 到 1.9.9）。

Microsoft CDN:

```
<head>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.0.js'
</script>
</head>
```

提示：使用谷歌或微软的 jQuery，有一个很大的优势：

许多用户在访问其他站点时，已经从谷歌或微软加载过 jQuery。所有结果是，当他们访问您的站点时，会从缓存中加载 jQuery，这样可以减少加载时间。同时，大多数 CDN 都可以确保当用户向其请求文件时，会从离用户最近的服务器上返回响应，这样也可以提高加载速度。

jQuery 语法

通过 **jQuery**，您可以选取（查询，**query**） **HTML** 元素，并对它们执行“操作”（**actions**）。

jQuery 语法实例

$(this).hide()

演示 jQuery hide() 函数，隐藏当前的 HTML 元素。

```
<html>
<head>
<script type="text/javascript" src="/jquery/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
  $(this).hide();
});
});
</script>
</head>

<body>
<button type="button">Click me</button>
</body>

</html>
```

$("#test").hide()

演示 jQuery hide() 函数，隐藏 id="test" 的元素。

```
<html>
<head>
<script type="text/javascript" src="/jquery/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p id="test">This is another paragraph.</p>
<button type="button">Click me</button>
</body>

</html>
```

$("p").hide()

演示 jQuery hide() 函数，隐藏所有

元素。

```
<html>
<head>
<script type="text/javascript" src="/jquery/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
$("button").click(function(){
$("p").hide();
});
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button type="button">Click me</button>
</body>
</html>
```

$(".test").hide()

演示 jQuery hide() 函数，隐藏所有 class="test" 的元素。

```
<html>
<head>
<script type="text/javascript" src="/jquery/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function()
{
  $("button").click(function()
  {
  $(".test").hide();
  });
});
</script>
</head>
<body>

<h2 class="test">This is a heading</h2>
<p class="test">This is a paragraph.</p>
<p>This is another paragraph.</p>
<button type="button">Click me</button>

</body>
</html>
```

jQuery 语法

jQuery 语法是为 HTML 元素的选取编制的，可以对元素执行某些操作。

基础语法是：*$(selector).action()*

- 美元符号定义 jQuery
- 选择符（selector）“查询”和“查找” HTML 元素
- jQuery 的 action() 执行对元素的操作

示例

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$(".test").hide() - 隐藏所有 class="test" 的所有元素

$("#test").hide() - 隐藏所有 id="test" 的元素

提示：jQuery 使用的语法是 XPath 与 CSS 选择器语法的组合。在本教程接下来的章节，您将学习到更多有关选择器的语法。

文档就绪函数

您也许已经注意到在我们的实例中的所有 jQuery 函数位于一个 document ready 函数中：

```
$(document).ready(function(){

--- jQuery functions go here ----

});
```

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

如果在文档没有完全加载之前就运行函数，操作可能失败。下面是两个具体的例子：

- 试图隐藏一个不存在的元素
- 获得未完全加载的图像的大小

jQuery 选择器

选择器允许您对元素组或单个元素进行操作。

jQuery 选择器

在前面的章节中，我们展示了一些有关如何选取 HTML 元素的实例。

关键点是学习 jQuery 选择器是如何准确地选取您希望应用效果的元素。

jQuery 元素选择器和属性选择器允许您通过标签名、属性名或内容对 HTML 元素进行选择。

选择器允许您对 HTML 元素组或单个元素进行操作。

在 HTML DOM 术语中：

选择器允许您对 DOM 元素组或单个 DOM 节点进行操作。

jQuery 元素选择器

jQuery 使用 CSS 选择器来选取 HTML 元素。

$("p") 选取 <p> 元素。

$("p.intro") 选取所有 class="intro" 的 <p> 元素。

$("p#demo") 选取所有 id="demo" 的 <p> 元素。

jQuery 属性选择器

jQuery 使用 XPath 表达式来选择带有给定属性的元素。

$("[href]") 选取所有带有 href 属性的元素。

$("[href='#']") 选取所有带有 href 值等于 "#" 的元素。

$("[href!='#']") 选取所有带有 href 值不等于 "#" 的元素。

$("[href$='.jpg']") 选取所有 href 值以 ".jpg" 结尾的元素。

jQuery CSS 选择器

jQuery CSS 选择器可用于改变 HTML 元素的 CSS 属性。

下面的例子把所有 p 元素的背景颜色更改为红色：

实例

```
$("p").css("background-color","red");
```

更多的选择器实例

语法	描述
$(this)	当前 HTML 元素
$("p")	所有 `<p>` 元素
$("p.intro")	所有 class="intro" 的 `<p>` 元素
$(".intro")	所有 class="intro" 的元素
$("#intro")	id="intro" 的元素
$("ul li:first")	每个 `<ul>` 的第一个 `<li>` 元素
$("[href$='.jpg']")	所有带有以 ".jpg" 结尾的属性值的 href 属性
$("div#intro .head")	id="intro" 的 `<div>` 元素中的所有 class="head" 的元素

如需完整的参考手册，请访问我们的 jQuery 选择器参考手册。

jQuery 事件

jQuery 是为事件处理特别设计的。

jQuery 事件函数

jQuery 事件处理方法是 jQuery 中的核心函数。

事件处理程序指的是当 HTML 中发生某些事件时所调用的方法。术语由事件"触发"（或"激发"）经常会被使用。

通常会把 jQuery 代码放到 <head>部分的事件处理方法中：

实例

```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  `$("button").click(function(){
    $("p").hide();
  });`
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>

</html>
```

在上面的例子中，当按钮的点击事件被触发时会调用一个函数：

```
$("button").click(function() {..some code... } )
```

该方法隐藏所有 <p> 元素：

```
$("p").hide();
```

单独文件中的函数

如果您的网站包含许多页面，并且您希望您的 jQuery 函数易于维护，那么请把您的 jQuery 函数放到独立的 .js 文件中。

当我们在教程中演示 jQuery 时，会将函数直接添加到 <head> 部分中。不过，把它们放到一个单独的文件中会更好，就像这样（通过 src 属性来引用文件）：

实例

```
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript" src="my_jquery_functions.js"></scrip
</head>
```

jQuery 名称冲突

jQuery 使用 $ 符号作为 jQuery 的简介方式。

某些其他 JavaScript 库中的函数（比如 Prototype）同样使用 $ 符号。

jQuery 使用名为 noConflict() 的方法来解决该问题。

var jq=jQuery.noConflict()，帮助您使用自己的名称（比如 jq）来代替 $ 符号。

结论

由于 jQuery 是为处理 HTML 事件而特别设计的，那么当您遵循以下原则时，您的代码会更恰当且更易维护：

- 把所有 jQuery 代码置于事件处理函数中
- 把所有事件处理函数置于文档就绪事件处理器中
- 把 jQuery 代码置于单独的 .js 文件中
- 如果存在名称冲突，则重命名 jQuery 库

jQuery 事件

下面是 jQuery 中事件方法的一些例子：

Event 函数	绑定函数至
$(document).ready(function)	将函数绑定到文档的就绪事件（当文档完成加载时）
$(selector).click(function)	触发或将函数绑定到被选元素的点击事件
$(selector).dblclick(function)	触发或将函数绑定到被选元素的双击事件
$(selector).focus(function)	触发或将函数绑定到被选元素的获得焦点事件
$(selector).mouseover(function)	触发或将函数绑定到被选元素的鼠标悬停事件

如需完整的参考手册，请访问我们的 jQuery 事件参考手册。

jQuery 效果

jQuery 效果 - 隐藏和显示

隐藏、显示、切换，滑动，淡入淡出，以及动画，哇哦！

实例

jQuery hide()

演示一个简单的 jQuery hide() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js">
</script>
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>
<p>如果您点击我，我会消失。</p>
<p>点击我，我会消失。</p>
<p>也要点击我哦。</p>
</body>
</html>
```

jQuery hide()

另一个 hide() 演示。如何隐藏部分文本。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $(".ex .hide").click(function(){
    $(this).parents(".ex").hide("slow");
  });
});
</script>
<style type="text/css">
div.ex
{
background-color:#e5eecc;
padding:7px;
border:solid 1px #c3c3c3;
}
</style>
</head>

<body>

<h3>中国办事处</h3>
<div class="ex">
<button class="hide" type="button">隐藏</button>
<p>联系人：张先生<br />
北三环中路 100 号<br />
北京</p>
</div>

<h3>美国办事处</h3>
<div class="ex">
<button class="hide" type="button">隐藏</button>
<p>联系人：David<br />
第五大街 200 号<br />
纽约</p>
</div>

</body>
</html>
```

jQuery hide() 和 show()

通过 jQuery，您可以使用 hide() 和 show() 方法来隐藏和显示 HTML 元素：

```
$("#hide").click(function(){
  $("p").hide();
});

$("#show").click(function(){
  $("p").show();
});
```

语法：

```
$(selector).hide(speed,callback);

$(selector).show(speed,callback);
```

可选的 speed 参数规定隐藏/显示的速度，可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是隐藏或显示完成后所执行的函数名称。

下面的例子演示了带有 speed 参数的 hide() 方法：

实例

```
$("button").click(function(){
  $("p").hide(1000);
});
```

jQuery toggle()

通过 jQuery，您可以使用 toggle() 方法来切换 hide() 和 show() 方法。

显示被隐藏的元素，并隐藏已显示的元素：

实例

```
$("button").click(function(){
  $("p").toggle();
});
```

语法：

```
$(selector).toggle(speed,callback);
```

可选的 speed 参数规定隐藏/显示的速度，可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是 toggle() 方法完成后所执行的函数名称。

jQuery 效果参考手册

如需全面查阅 jQuery 效果，请访问我们的 jQuery 效果参考手册。

jQuery 效果 - 淡入淡出

通过 **jQuery**，您可以实现元素的淡入淡出效果。

实例

jQuery fadeIn()

演示 jQuery fadeIn() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
  });
});
</script>
</head>

<body>
<p>演示带有不同参数的 fadeIn() 方法。</p>
<button>点击这里，使三个矩形淡入</button>
<br><br>
<div id="div1" style="width:80px;height:80px;display:none;backgrour
<br>
<div id="div2" style="width:80px;height:80px;display:none;backgrour
<br>
<div id="div3" style="width:80px;height:80px;display:none;backgrour
</body>
</html>
```

jQuery fadeOut()

演示 jQuery fadeOut() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
  });
});
</script>
</head>

<body>
<p>演示带有不同参数的 fadeOut() 方法。</p>
<button>点击这里，使三个矩形淡出</button>
<br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;'
<br>
<div id="div2" style="width:80px;height:80px;background-color:greer
<br>
<div id="div3" style="width:80px;height:80px;background-color:blue,
</body>

</html>
```

jQuery fadeToggle()

演示 jQuery fadeToggle() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeToggle();
    $("#div2").fadeToggle("slow");
    $("#div3").fadeToggle(3000);
  });
});
</script>
</head>

<body>

<p>演示带有不同参数的 fadeToggle() 方法。</p>
<button>点击这里，使三个矩形淡入淡出</button>
<br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;'
<br>
<div id="div2" style="width:80px;height:80px;background-color:greer
<br>
<div id="div3" style="width:80px;height:80px;background-color:blue,
</body>

</body>
</html>
```

jQuery fadeTo()

演示 jQuery fadeTo() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeTo("slow",0.15);
    $("#div2").fadeTo("slow",0.4);
    $("#div3").fadeTo("slow",0.7);
  });
});
</script>
</head>

<body>

<p>演示带有不同参数的 fadeTo() 方法。</p>
<button>点击这里，使三个矩形淡出</button>
<br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;'
<br>
<div id="div2" style="width:80px;height:80px;background-color:greer
<br>
<div id="div3" style="width:80px;height:80px;background-color:blue;

</body>
</html>
```

jQuery Fading 方法

通过 jQuery，您可以实现元素的淡入淡出效果。

jQuery 拥有下面四种 fade 方法：

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

jQuery fadeIn() 方法

jQuery fadeIn() 用于淡入已隐藏的元素。

语法：

```
$(selector).fadeIn(speed,callback);
```

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeIn() 方法：

实例

```
$("button").click(function(){
  $("#div1").fadeIn();
  $("#div2").fadeIn("slow");
  $("#div3").fadeIn(3000);
});
```

jQuery fadeOut() 方法

jQuery fadeOut() 方法用于淡出可见元素。

语法：

```
$(selector).fadeOut(speed,callback);
```

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeOut() 方法：

实例

```
$("button").click(function(){
  $("#div1").fadeOut();
  $("#div2").fadeOut("slow");
  $("#div3").fadeOut(3000);
});
```

jQuery fadeToggle() 方法

jQuery fadeToggle() 方法可以在 fadeIn() 与 fadeOut() 方法之间进行切换。

如果元素已淡出，则 fadeToggle() 会向元素添加淡入效果。

如果元素已淡入，则 fadeToggle() 会向元素添加淡出效果。

语法：

```
$(selector).fadeToggle(speed,callback);
```

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeToggle() 方法：

实例

```
$("button").click(function(){
  $("#div1").fadeToggle();
  $("#div2").fadeToggle("slow");
  $("#div3").fadeToggle(3000);
});
```

jQuery fadeTo() 方法

jQuery fadeTo() 方法允许渐变为给定的不透明度（值介于 0 与 1 之间）。

语法：

```
$(selector).fadeTo(speed,opacity,callback);
```

必需的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

fadeTo() 方法中必需的 opacity 参数将淡入淡出效果设置为给定的不透明度（值介于 0 与 1 之间）。

可选的 callback 参数是该函数完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeTo() 方法：

实例

```
$("button").click(function(){
  $("#div1").fadeTo("slow",0.15);
  $("#div2").fadeTo("slow",0.4);
  $("#div3").fadeTo("slow",0.7);
});
```

jQuery 效果参考手册

如需全面查阅 jQuery 效果，请访问我们的 jQuery 效果参考手册。

jQuery 效果 - 滑动

jQuery 滑动方法可使元素上下滑动。

实例

jQuery slideDown()

演示 jQuery slideDown() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $(".flip").click(function(){
    $(".panel").slideDown("slow");
  });
});
</script>

<style type="text/css">
div.panel,p.flip
{
margin:0px;
padding:5px;
text-align:center;
background:#e5eecc;
border:solid 1px #c3c3c3;
}
div.panel
{
height:120px;
display:none;
}
</style>
</head>

<body>

<div class="panel">
<p>W3School - 领先的 Web 技术教程站点</p>
<p>在 W3School，你可以找到你所需要的所有网站建设教程。</p>
</div>

<p class="flip">请点击这里</p>

</body>
</html>
```

jQuery slideUp()

演示 jQuery slideUp() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $(".flip").click(function(){
    $(".panel").slideUp("slow");
  });
});
</script>

<style type="text/css">
div.panel,p.flip
{
margin:0px;
padding:5px;
text-align:center;
background:#e5eecc;
border:solid 1px #c3c3c3;
}
div.panel
{
height:120px;
}
</style>
</head>

<body>

<div class="panel">
<p>W3School - 领先的 Web 技术教程站点</p>
<p>在 W3School，你可以找到你所需要的所有网站建设教程。</p>
</div>

<p class="flip">请点击这里</p>

</body>
</html>
```

jQuery slideToggle()

演示 jQuery slideToggle() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
$(".flip").click(function(){
    $(".panel").slideToggle("slow");
  });
});
</script>

<style type="text/css">
div.panel,p.flip
{
margin:0px;
padding:5px;
text-align:center;
background:#e5eecc;
border:solid 1px #c3c3c3;
}
div.panel
{
height:120px;
display:none;
}
</style>
</head>

<body>

<div class="panel">
<p>W3School - 领先的 Web 技术教程站点</p>
<p>在 W3School，你可以找到你所需要的所有网站建设教程。</p>
</div>

<p class="flip">请点击这里</p>

</body>
</html>
```

jQuery 滑动方法

通过 jQuery，您可以在元素上创建滑动效果。

jQuery 拥有以下滑动方法：

- slideDown()
- slideUp()
- slideToggle()

jQuery slideDown() 方法

jQuery slideDown() 方法用于向下滑动元素。

语法：

```
$(selector).slideDown(speed,callback);
```

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideDown() 方法：

实例

```
$("#flip").click(function(){
  $("#panel").slideDown();
});
```

jQuery slideUp() 方法

jQuery slideUp() 方法用于向上滑动元素。

语法：

```
$(selector).slideUp(speed,callback);
```

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideUp() 方法：

实例

```
$("#flip").click(function(){
  $("#panel").slideUp();
});
```

jQuery slideToggle() 方法

jQuery slideToggle() 方法可以在 slideDown() 与 slideUp() 方法之间进行切换。

如果元素向下滑动，则 slideToggle() 可向上滑动它们。

如果元素向上滑动，则 slideToggle() 可向下滑动它们。

```
$(selector).slideToggle(speed,callback);
```

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideToggle() 方法：

实例

```
$("#flip").click(function(){
  $("#panel").slideToggle();
});
```

jQuery 效果参考手册

如需全面查阅 jQuery 效果，请访问我们的 jQuery 效果参考手册。

jQuery 效果 - 动画

jQuery animate() 方法允许您创建自定义的动画。

jQuery 动画 - animate() 方法

jQuery animate() 方法用于创建自定义动画。

语法：

```
$(selector).animate({params},speed,callback);
```

必需的 params 参数定义形成动画的 CSS 属性。

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是动画完成后所执行的函数名称。

下面的例子演示 animate() 方法的简单应用；它把 <div> 元素移动到左边，直到 left 属性等于 250 像素为止：

实例

```
$("button").click(function(){
  $("div").animate({left:'250px'});
});
```

提示：默认地，所有 HTML 元素都有一个静态位置，且无法移动。

如需对位置进行操作，要记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute！

jQuery animate() - 操作多个属性

请注意，生成动画的过程中可同时使用多个属性：

实例

```
$("button").click(function(){
  $("div").animate({
    left:'250px',
    opacity:'0.5',
    height:'150px',
    width:'150px'
  });
});
```

提示：可以用 animate() 方法来操作所有 CSS 属性吗？

是的，几乎可以！不过，需要记住一件重要的事情：当使用 animate() 时，必须使用 Camel 标记法书写所有的属性名，比如，必须使用 paddingLeft 而不是 padding-left，使用 marginRight 而不是 margin-right，等等。

同时，色彩动画并不包含在核心 jQuery 库中。

如果需要生成颜色动画，您需要从 jQuery.com 下载 Color Animations 插件。

jQuery animate() - 使用相对值

也可以定义相对值（该值相对于元素的当前值）。需要在值的前面加上 += 或 -=：

实例

```
$("button").click(function(){
  $("div").animate({
    left:'250px',
    height:'+=150px',
    width:'+=150px'
  });
});
```

jQuery animate() - 使用预定义的值

您甚至可以把属性的动画值设置为 "show"、"hide" 或 "toggle"：

实例

```
$("button").click(function(){
  $("div").animate({
    height:'toggle'
  });
});
```

jQuery animate() - 使用队列功能

默认地，jQuery 提供针对动画的队列功能。

这意味着如果您在彼此之后编写多个 animate() 调用，jQuery 会创建包含这些方法调用的“内部”队列。然后逐一运行这些 animate 调用。

实例 1

隐藏，如果您希望在彼此之后执行不同的动画，那么我们要利用队列功能：

```
$("button").click(function(){
  var div=$("div");
  div.animate({height:'300px',opacity:'0.4'},"slow");
  div.animate({width:'300px',opacity:'0.8'},"slow");
  div.animate({height:'100px',opacity:'0.4'},"slow");
  div.animate({width:'100px',opacity:'0.8'},"slow");
});
```

实例 2

下面的例子把 <div> 元素移动到右边，然后增加文本的字号：

```
$("button").click(function(){
  var div=$("div");
  div.animate({left:'100px'},"slow");
  div.animate({fontSize:'3em'},"slow");
});
```

jQuery 停止动画

jQuery stop() 方法用于在动画或效果完成前对它们进行停止。

实例

jQuery stop() 滑动

演示 jQuery stop() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideDown(5000);
  });
  $("#stop").click(function(){
    $("#panel").stop();
  });
});
</script>

<style type="text/css">
#panel,#flip
{
padding:5px;
text-align:center;
background-color:#e5eecc;
border:solid 1px #c3c3c3;
}
#panel
{
padding:50px;
display:none;
}
</style>
</head>

<body>

<button id="stop">停止滑动</button>
<div id="flip">点击这里，向下滑动面板</div>
<div id="panel">Hello world!</div>

</body>
</html>
```

jQuery stop() 动画（带有参数）

演示 jQuery stop() 方法。

```
<!DOCTYPE html>
<html>
<head>
<script src="/jquery/jquery-1.11.1.min.js">
</script>
<script>
$(document).ready(function(){
  $("#start").click(function(){
    $("div").animate({left:'100px'},5000);
    $("div").animate({fontSize:'3em'},5000);
  });

  $("#stop").click(function(){
    $("div").stop();
  });

  $("#stop2").click(function(){
    $("div").stop(true);
  });

  $("#stop3").click(function(){
    $("div").stop(true,true);
  });

});
</script>
</head>
<body>

<button id="start">开始</button>
<button id="stop">停止</button>
<button id="stop2">停止所有</button>
<button id="stop3">停止但要完成</button>
<p><b>"开始"</b> 按钮会启动动画。</p>
<p><b>"停止"</b> 按钮会停止当前活动的动画，但允许已排队的动画向前执行。</p>
<p><b>"停止所有"</b> 按钮停止当前活动的动画，并清空动画队列；因此元素上的所有
<p><b>"停止但要完成"</b> 会立即完成当前活动的动画，然后停下来。</p>

<div style="background:#98bf21;height:100px;width:200px;position:ab

</body>
</html>
```

jQuery stop() 方法

jQuery stop() 方法用于停止动画或效果，在它们完成之前。

stop() 方法适用于所有 jQuery 效果函数，包括滑动、淡入淡出和自定义动画。

语法

```
$(selector).stop(stopAll,goToEnd);
```

可选的 stopAll 参数规定是否应该清除动画队列。默认是 false，即仅停止活动的动画，允许任何排入队列的动画向后执行。

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

下面的例子演示 stop() 方法，不带参数：

实例

```
$("#stop").click(function(){
  $("#panel").stop();
});
```

jQuery 效果参考手册

如需全面查阅 jQuery 效果，请访问我们的 jQuery 效果参考手册。

jQuery Callback 函数

Callback 函数在当前动画 **100%** 完成之后执行。

jQuery 动画的问题

许多 jQuery 函数涉及动画。这些函数也许会将 *speed* 或 *duration* 作为可选参数。

例子：*$("p").hide("slow")*

speed 或 *duration* 参数可以设置许多不同的值，比如 "slow", "fast", "normal" 或毫秒。

实例

```
$("button").click(function(){
$("p").hide(1000);
});
```

由于 JavaScript 语句（指令）是逐一执行的 - 按照次序，动画之后的语句可能会产生错误或页面冲突，因为动画还没有完成。

为了避免这个情况，您可以以参数的形式添加 Callback 函数。

jQuery Callback 函数

当动画 100% 完成后，即调用 Callback 函数。

典型的语法：

```
$(selector).hide(speed,callback)
```

callback 参数是一个在 hide 操作完成后被执行的函数。

错误（没有 **callback**）

```
$("p").hide(1000);
alert("The paragraph is now hidden");
```

正确（有 callback）

```
$("p").hide(1000,function(){
alert("The paragraph is now hidden");
});
```

结论：如果您希望在一个涉及动画的函数之后来执行语句，请使用 callback 函数。

jQuery - Chaining

通过 **jQuery**，您可以把动作/方法链接起来。

Chaining 允许我们在一条语句中允许多个 **jQuery** 方法（在相同的元素上）。

jQuery 方法链接

直到现在，我们都是一次写一条 jQuery 语句（一条接着另一条）。

不过，有一种名为链接（chaining）的技术，允许我们在相同的元素上运行多条 jQuery 命令，一条接着另一条。

提示：这样的话，浏览器就不必多次查找相同的元素。

如需链接一个动作，您只需简单地把该动作追加到之前的动作上。

例子 1

下面的例子把 css(), slideUp(), and slideDown() 链接在一起。"p1" 元素首先会变为红色，然后向上滑动，然后向下滑动：

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

如果需要，我们也可以添加多个方法调用。

提示：当进行链接时，代码行会变得很差。不过，jQuery 在语法上不是很严格；您可以按照希望的格式来写，包含折行和缩进。

例子 2

这样写也可以运行：

```
$("#p1").css("color","red")
  .slideUp(2000)
  .slideDown(2000);
```

jQuery 会抛掉多余的空格，并按照一行长代码来执行上面的代码行。

jQuery HTML

jQuery - 获得内容和属性

jQuery 拥有可操作 **HTML** 元素和属性的强大方法。

jQuery DOM 操作

jQuery 中非常重要的部分，就是操作 DOM 的能力。

jQuery 提供一系列与 DOM 相关的方法，这使访问和操作元素和属性变得很容易。

提示：DOM = Document Object Model（文档对象模型）

DOM 定义访问 HTML 和 XML 文档的标准：

“W3C 文档对象模型独立于平台和语言的界面，允许程序和脚本动态访问和更新文档的内容、结构以及样式。”

获得内容 - text()、html() 以及 val()

三个简单实用的用于 DOM 操作的 jQuery 方法：

- text() - 设置或返回所选元素的文本内容
- html() - 设置或返回所选元素的内容（包括 HTML 标记）
- val() - 设置或返回表单字段的值

下面的例子演示如何通过 jQuery text() 和 html() 方法来获得内容：

实例

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

下面的例子演示如何通过 jQuery val() 方法获得输入字段的值：

实例

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```

获取属性 - attr()

jQuery attr() 方法用于获取属性值。

下面的例子演示如何获得链接中 href 属性的值：

实例

```
$("button").click(function(){
  alert($("#w3s").attr("href"));
});
```

下一章会讲解如何设置（改变）内容和属性值。

jQuery HTML 参考手册

如需有关 jQuery HTML 方法的完整内容，请访问以下参考手册：

jQuery - 设置内容和属性

设置内容 - text()、html() 以及 val()

我们将使用前一章中的三个相同的方法来设置内容：

- text() - 设置或返回所选元素的文本内容
- html() - 设置或返回所选元素的内容（包括 HTML 标记）
- val() - 设置或返回表单字段的值

下面的例子演示如何通过 text()、html() 以及 val() 方法来设置内容：

实例

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
  $("#test3").val("Dolly Duck");
});
```

text()、html() 以及 val() 的回调函数

上面的三个 jQuery 方法：text()、html() 以及 val()，同样拥有回调函数。回调函数由两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回您希望使用的字符串。

下面的例子演示带有回调函数的 text() 和 html()：

实例

```
$("#btn1").click(function(){
  $("#test1").text(function(i,origText){
    return "Old text: " + origText + " New text: Hello world!
    (index: " + i + ")";
  });
});

$("#btn2").click(function(){
  $("#test2").html(function(i,origText){
    return "Old html: " + origText + " New html: Hello <b>world!</b
    (index: " + i + ")";
  });
});
```

设置属性 - attr()

jQuery attr() 方法也用于设置/改变属性值。

下面的例子演示如何改变（设置）链接中 href 属性的值：

实例

```
$("button").click(function(){
  $("#w3s").attr("href","http://www.w3school.com.cn/jquery");
});
```

attr() 方法也允许您同时设置多个属性。

下面的例子演示如何同时设置 href 和 title 属性：

实例

```
$("button").click(function(){
  $("#w3s").attr({
    "href" : "http://www.w3school.com.cn/jquery",
    "title" : "W3School jQuery Tutorial"
  });
});
```

attr() 的回调函数

jQuery 方法 attr()，也提供回调函数。回调函数由两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回您希望使用的字符串。

下面的例子演示带有回调函数的 attr() 方法：

实例

```
$("button").click(function(){
  $("#w3s").attr("href", function(i,origValue){
    return origValue + "/jquery";
  });
});
```

jQuery HTML 参考手册

如需有关 jQuery HTML 方法的完整内容，请访问以下参考手册：

jQuery - 添加元素

通过 **jQuery**，可以很容易地添加新元素/内容。

添加新的 HTML 内容

我们将学习用于添加新内容的四个 jQuery 方法：

- append() - 在被选元素的结尾插入内容
- prepend() - 在被选元素的开头插入内容
- after() - 在被选元素之后插入内容
- before() - 在被选元素之前插入内容

jQuery append() 方法

jQuery append() 方法在被选元素的结尾插入内容。

实例

```
$("p").append("Some appended text.");
```

jQuery prepend() 方法

jQuery prepend() 方法在被选元素的开头插入内容。

实例

```
$("p").prepend("Some prepended text.");
```

通过 append() 和 prepend() 方法添加若干新元素

在上面的例子中，我们只在被选元素的开头/结尾插入文本/HTML。

不过，append() 和 prepend() 方法能够通过参数接收无限数量的新元素。可以通过 jQuery 来生成文本/HTML（就像上面的例子那样），或者通过 JavaScript 代码和 DOM 元素。

在下面的例子中，我们创建若干个新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 append() 方法把这些新元素追加到文本中（对 prepend() 同样有效）：

实例

```
function appendText()
{
var txt1="<p>Text.</p>";               // 以 HTML 创建新元素
var txt2=$("<p></p>").text("Text.");   // 以 jQuery 创建新元素
var txt3=document.createElement("p");  // 以 DOM 创建新元素
txt3.innerHTML="Text.";
$("p").append(txt1,txt2,txt3);         // 追加新元素
}
```

jQuery after() 和 before() 方法

jQuery after() 方法在被选元素之后插入内容。

jQuery before() 方法在被选元素之前插入内容。

实例

```
$("img").after("Some text after");

$("img").before("Some text before");
```

通过 after() 和 before() 方法添加若干新元素

after() 和 before() 方法能够通过参数接收无限数量的新元素。可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建新元素。

在下面的例子中，我们创建若干新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 after() 方法把这些新元素插到文本中（对 before() 同样有效）：

实例

```
function afterText()
{
var txt1="<b>I </b>";                        // 以 HTML 创建新元素
var txt2=$("<i></i>").text("love ");         // 通过 jQuery 创建新元素
var txt3=document.createElement("big");  // 通过 DOM 创建新元素
txt3.innerHTML="jQuery!";
$("img").after(txt1,txt2,txt3);              // 在 img 之后插入新元素
}
```

jQuery HTML 参考手册

如需有关 jQuery HTML 方法的完整内容，请访问以下参考手册：

jQuery - 删除元素

通过 **jQuery**，可以很容易地删除已有的 **HTML** 元素。

删除元素/内容

如需删除元素和内容，一般可使用以下两个 jQuery 方法：

- remove() - 删除被选元素（及其子元素）
- empty() - 从被选元素中删除子元素

jQuery remove() 方法

jQuery remove() 方法删除被选元素及其子元素。

实例

```
$("#div1").remove();
```

jQuery empty() 方法

jQuery empty() 方法删除被选元素的子元素。

实例

```
$("#div1").empty();
```

过滤被删除的元素

jQuery remove() 方法也可接受一个参数，允许您对被删元素进行过滤。

该参数可以是任何 jQuery 选择器的语法。

下面的例子删除 class="italic" 的所有 <p> 元素：

实例

```
$("p").remove(".italic");
```

jQuery HTML 参考手册

如需有关 jQuery HTML 方法的完整内容，请访问以下参考手册：

jQuery - 获取并设置 CSS 类

通过 **jQuery**，可以很容易地对 **CSS** 元素进行操作。

切换类

jQuery 操作 CSS

jQuery 拥有若干进行 CSS 操作的方法。我们将学习下面这些：

- addClass() - 向被选元素添加一个或多个类
- removeClass() - 从被选元素删除一个或多个类
- toggleClass() - 对被选元素进行添加/删除类的切换操作
- css() - 设置或返回样式属性

实例样式表

下面的样式表将用于本页的所有例子：

```
.important
{
font-weight:bold;
font-size:xx-large;
}

.blue
{
color:blue;
}
```

jQuery addClass() 方法

下面的例子展示如何向不同的元素添加 class 属性。当然，在添加类时，您也可以选取多个元素：

实例

```
$("button").click(function(){
  $("h1,h2,p").addClass("blue");
  $("div").addClass("important");
});
```

您也可以在 addClass() 方法中规定多个类：

实例

```
$("button").click(function(){
  $("#div1").addClass("important blue");
});
```

jQuery removeClass() 方法

下面的例子演示如何不同的元素中删除指定的 class 属性：

实例

```
$("button").click(function(){
  $("h1,h2,p").removeClass("blue");
});
```

jQuery toggleClass() 方法

下面的例子将展示如何使用 jQuery toggleClass() 方法。该方法对被选元素进行添加/删除类的切换操作：

实例

```
$("button").click(function(){
  $("h1,h2,p").toggleClass("blue");
});
```

jQuery css() 方法

我们将在下一章讲解 jQuery css() 方法。

jQuery HTML 参考手册

如需有关 jQuery CSS 方法的完整内容，请访问我们的 jQuery CSS 操作参考手册

jQuery - css() 方法

jQuery css() 方法

css() 方法设置或返回被选元素的一个或多个样式属性。

返回 CSS 属性

如需返回指定的 CSS 属性的值，请使用如下语法：

```
css("propertyname");
```

下面的例子将返回首个匹配元素的 background-color 值：

实例

```
$("p").css("background-color");
```

设置 CSS 属性

如需设置指定的 CSS 属性，请使用如下语法：

```
css("propertyname","value");
```

下面的例子将为所有匹配元素设置 background-color 值：

实例

```
$("p").css("background-color","yellow");
```

设置多个 CSS 属性

如需设置多个 CSS 属性，请使用如下语法：

```
css({"propertyname":"value","propertyname":"value",...});
```

下面的例子将为所有匹配元素设置 background-color 和 font-size：

实例

```
$("p").css({"background-color":"yellow","font-size":"200%"});
```

jQuery HTML 参考手册

如需有关 jQuery CSS 方法的完整内容，请访问我们的 jQuery CSS 操作参考手册

jQuery - 尺寸

通过 **jQuery**，很容易处理元素和浏览器窗口的尺寸。

jQuery 尺寸 方法

jQuery 提供多个处理尺寸的重要方法：

- width()
- height()
- innerWidth()
- innerHeight()
- outerWidth()
- outerHeight()

jQuery width() 和 height() 方法

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

下面的例子返回指定的 <div> 元素的宽度和高度：

实例

```
$("button").click(function(){
  var txt="";
  txt+="Width: " + $("#div1").width() + "</br>";
  txt+="Height: " + $("#div1").height();
  $("#div1").html(txt);
});
```

jQuery innerWidth() 和 innerHeight() 方法

innerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

下面的例子返回指定的 <div> 元素的 inner-width/height：

实例

```
$("button").click(function(){
  var txt="";
  txt+="Inner width: " + $("#div1").innerWidth() + "</br>";
  txt+="Inner height: " + $("#div1").innerHeight();
  $("#div1").html(txt);
});
```

jQuery outerWidth() 和 outerHeight() 方法

outerWidth() 方法返回元素的宽度（包括内边距和边框）。

outerHeight() 方法返回元素的高度（包括内边距和边框）。

下面的例子返回指定的 <div> 元素的 outer-width/height：

实例

```
$("button").click(function(){
  var txt="";
  txt+="Outer width: " + $("#div1").outerWidth() + "</br>";
  txt+="Outer height: " + $("#div1").outerHeight();
  $("#div1").html(txt);
});
```

outerWidth(true) 方法返回元素的宽度（包括内边距、边框和外边距）。

outerHeight(true) 方法返回元素的高度（包括内边距、边框和外边距）。

实例

```
$("button").click(function(){
  var txt="";
  txt+="Outer width (+margin): " + $("#div1").outerWidth(true) + "<
  txt+="Outer height (+margin): " + $("#div1").outerHeight(true);
  $("#div1").html(txt);
});
```

jQuery - 更多的 width() 和 height()

下面的例子返回文档（HTML 文档）和窗口（浏览器视口）的宽度和高度：

实例

```
$("button").click(function(){
  var txt="";
  txt+="Document width/height: " + $(document).width();
  txt+="x" + $(document).height() + "\n";
  txt+="Window width/height: " + $(window).width();
  txt+="x" + $(window).height();
  alert(txt);
});
```

下面的例子设置指定的 <div> 元素的宽度和高度：

实例

```
$("button").click(function(){
  $("#div1").width(500).height(500);
});
```

jQuery CSS 参考手册

如需关于 jQuery Dimensions 的完整参考，请访问我们的 jQuery 尺寸参考手册。

jQuery 遍历

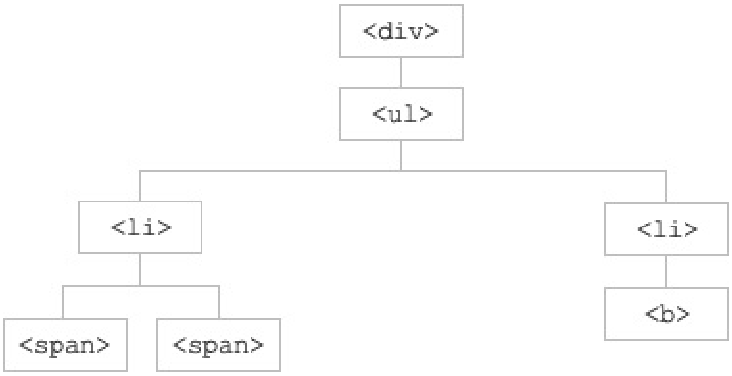
jQuery 遍历

什么是遍历？

jQuery 遍历，意为"移动"，用于根据其相对于其他元素的关系来"查找"（或选取）HTML 元素。以某项选择开始，并沿着这个选择移动，直到抵达您期望的元素为止。

下图展示了一个家族树。通过 jQuery 遍历，您能够从被选（当前的）元素开始，轻松地在家族树中向上移动（祖先），向下移动（子孙），水平移动（同胞）。这种移动被称为对 DOM 进行遍历。

图示解释：



- `<div>` 元素是 `<ul>` 的父元素，同时是其中所有内容的祖先。
- `<ul>` 元素是 `<li>` 元素的父元素，同时是 `<div>` 的子元素
- 左边的 `<li>` 元素是 `<span>` 的父元素， `<ul>` 的子元素，同时是 `<div>` 的后代。
- `<span>` 元素是 `<li>` 的子元素，同时是 `<ul>` 和 `<div>` 的后代。
- 两个 `<li>` 元素是同胞（拥有相同的父元素）。
- 右边的 `<li>` 元素是 `<b>` 的父元素， `<ul>` 的子元素，同时是 `<div>` 的后代。
- `<b>` 元素是右边的 `<li>` 的子元素，同时是 `<ul>` 和 `<div>` 的后代。

提示：祖先是父、祖父、曾祖父等等。后代是子、孙、曾孙等等。同胞拥有相同的父。

遍历 DOM

jQuery 提供了多种遍历 DOM 的方法。

遍历方法中最大的种类是树遍历（tree-traversal）。

下一章会讲解如何在 DOM 树中向上、下以及同级移动。

jQuery 遍历参考手册

如需了解所有的 jQuery 遍历方法，请访问我们的 jQuery 遍历参考手册。

jQuery 遍历 - 祖先

祖先是父、祖父或曾祖父等等。

通过 jQuery，您能够向上遍历 DOM 树，以查找元素的祖先。

向上遍历 DOM 树

这些 jQuery 方法很有用，它们用于向上遍历 DOM 树：

- parent()
- parents()
- parentsUntil()

jQuery parent() 方法

parent() 方法返回被选元素的直接父元素。

该方法只会向上一级对 DOM 树进行遍历。

下面的例子返回每个 <span> 元素的的直接父元素：

实例

```
$(document).ready(function(){
  $("span").parent();
});
```

jQuery parents() 方法

parents() 方法返回被选元素的所有祖先元素，它一路向上直到文档的根元素 (<html>)。

下面的例子返回所有 <span> 元素的所有祖先：

实例

```
$(document).ready(function(){
  $("span").parents();
});
```

您也可以使用可选参数来过滤对祖先元素的搜索。

下面的例子返回所有 <span> 元素的所有祖先，并且它是 <ul> 元素：

实例

```
$(document).ready(function(){
  $("span").parents("ul");
});
```

jQuery parentsUntil() 方法

parentsUntil() 方法返回介于两个给定元素之间的所有祖先元素。

下面的例子返回介于 <span> 与 <div> 元素之间的所有祖先元素：

实例

```
$(document).ready(function(){
  $("span").parentsUntil("div");
});
```

jQuery 遍历参考手册

如需了解所有的 jQuery 遍历方法，请访问我们的 jQuery 遍历参考手册。

jQuery 遍历 - 后代

后代是子、孙、曾孙等等。

通过 jQuery，您能够向下遍历 DOM 树，以查找元素的后代。

向下遍历 DOM 树

下面是两个用于向下遍历 DOM 树的 jQuery 方法：

- children()
- find()

jQuery children() 方法

children() 方法返回被选元素的所有直接子元素。

该方法只会向下一级对 DOM 树进行遍历。

下面的例子返回每个 <div> 元素的所有直接子元素：

实例

```
$(document).ready(function(){
  $("div").children();
});
```

您也可以使用可选参数来过滤对子元素的搜索。

下面的例子返回类名为 "1" 的所有 <p> 元素，并且它们是 <div> 的直接子元素：

实例

```
$(document).ready(function(){
  $("div").children("p.1");
});
```

jQuery find() 方法

find() 方法返回被选元素的后代元素，一路向下直到最后一个后代。

下面的例子返回属于 <div> 后代的所有 <span> 元素：

实例

```
$(document).ready(function(){
  $("div").find("span");
});
```

下面的例子返回 <div> 的所有后代：

实例

```
$(document).ready(function(){
  $("div").find("*");
});
```

jQuery 遍历参考手册

如需了解所有的 jQuery 遍历方法，请访问我们的 jQuery 遍历参考手册。

jQuery 遍历 - 同胞

同胞拥有相同的父元素。

通过 jQuery，您能够在 DOM 树中遍历元素的同胞元素。

在 DOM 树中水平遍历

有许多有用的方法让我们在 DOM 树进行水平遍历：

- siblings()
- next()
- nextAll()
- nextUntil()
- prev()
- prevAll()
- prevUntil()

jQuery siblings() 方法

siblings() 方法返回被选元素的所有同胞元素。

下面的例子返回 <h2> 的所有同胞元素：

实例

```
$(document).ready(function(){
  $("h2").siblings();
});
```

您也可以使用可选参数来过滤对同胞元素的搜索。

下面的例子返回属于 <h2> 的同胞元素的所有 <p> 元素：

实例

```
$(document).ready(function(){
  $("h2").siblings("p");
});
```

jQuery next() 方法

next() 方法返回被选元素的下一个同胞元素。

该方法只返回一个元素。

下面的例子返回 <h2> 的下一个同胞元素：

实例

```
$(document).ready(function(){
  $("h2").next();
});
```

jQuery nextAll() 方法

nextAll() 方法返回被选元素的所有跟随的同胞元素。

下面的例子返回 <h2> 的所有跟随的同胞元素：

实例

```
$(document).ready(function(){
  $("h2").nextAll();
});
```

jQuery nextUntil() 方法

nextUntil() 方法返回介于两个给定参数之间的所有跟随的同胞元素。

下面的例子返回介于 <h2> 与 <h6> 元素之间的所有同胞元素：

实例

```
$(document).ready(function(){
  $("h2").nextUntil("h6");
});
```

jQuery prev(), prevAll() & prevUntil() 方法

prev(), prevAll() 以及 prevUntil() 方法的工作方式与上面的方法类似，只不过方向相反而已：它们返回的是前面的同胞元素（在 DOM 树中沿着同胞元素向后遍历，而不是向前）。

jQuery 遍历参考手册

如需了解所有的 jQuery 遍历方法，请访问我们的 jQuery 遍历参考手册。

jQuery 遍历 - 过滤

缩写搜索元素的范围

三个最基本的过滤方法是：first(), last() 和 eq()，它们允许您基于其在一组元素中的位置来选择一个特定的元素。

其他过滤方法，比如 filter() 和 not() 允许您选取匹配或不匹配某项指定标准的元素。

jQuery first() 方法

first() 方法返回被选元素的首个元素。

下面的例子选取首个 <div> 元素内部的第一个 <p> 元素：

实例

```
$(document).ready(function(){
  $("div p").first();
});
```

jQuery last() 方法

last() 方法返回被选元素的最后一个元素。

下面的例子选择最后一个 <div> 元素中的最后一个 <p> 元素：

实例

```
$(document).ready(function(){
  $("div p").last();
});
```

jQuery eq() 方法

eq() 方法返回被选元素中带有指定索引号的元素。

索引号从 0 开始，因此首个元素的索引号是 0 而不是 1。下面的例子选取第二个 <p> 元素（索引号 1）：

实例

```
$(document).ready(function(){
  $("p").eq(1);
});
```

jQuery filter() 方法

filter() 方法允许您规定一个标准。不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回。

下面的例子返回带有类名 "intro" 的所有 <p> 元素：

实例

```
$(document).ready(function(){
  $("p").filter(".intro");
});
```

jQuery not() 方法

not() 方法返回不匹配标准的所有元素。

提示：not() 方法与 filter() 相反。

下面的例子返回不带有类名 "intro" 的所有 <p> 元素：

实例

```
$(document).ready(function(){
  $("p").not(".intro");
});
```

jQuery 遍历参考手册

如需了解所有的 jQuery 遍历方法，请访问我们的 jQuery 遍历参考手册。

jQuery - AJAX

jQuery - AJAX 简介

AJAX 是与服务器交换数据的艺术，它在不重载全部页面的情况下，实现了对部分网页的更新。

jQuery AJAX 实例

```
<script>$(document).ready(function(){ $("#btn1").click(function(){

<h3>请点击下面的按钮，通过 jQuery AJAX 改变这段文本。</h3>

<button id="btn1" type="button">获得外部的内容</button>
```

什么是 AJAX？

AJAX = 异步 JavaScript 和 XML（Asynchronous JavaScript and XML）。

简短地说，在不重载整个网页的情况下，AJAX 通过后台加载数据，并在网页上进行显示。

使用 AJAX 的应用程序案例：谷歌地图、腾讯微博、优酷视频、人人网等等。

您可以在我们的 AJAX 教程中学到更多有关 AJAX 的知识。

关于 jQuery 与 AJAX

jQuery 提供多个与 AJAX 有关的方法。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

提示：如果没有 jQuery，AJAX 编程还是有些难度的。

编写常规的 AJAX 代码并不容易，因为不同的浏览器对 AJAX 的实现并不相同。这意味着您必须编写额外的代码对浏览器进行测试。不过，jQuery 团队为我们解决了这个难题，我们只需要一行简单的代码，就可以实现 AJAX 功能。

jQuery AJAX 方法

在下面的章节，我们将学习到最重要的 jQuery AJAX 方法。

jQuery - AJAX load() 方法

jQuery load() 方法

jQuery load() 方法是简单但强大的 AJAX 方法。

load() 方法从服务器加载数据，并把返回的数据放入被选元素中。

语法：

```
$(_selector_).load(_URL_,_data_,_callback_);
```

必需的 *URL* 参数规定您希望加载的 URL。

可选的 *data* 参数规定与请求一同发送的查询字符串键/值对集合。

可选的 *callback* 参数是 load() 方法完成后所执行的函数名称。

这是示例文件（"demo_test.txt"）的内容：

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```

下面的例子会把文件 "demo_test.txt" 的内容加载到指定的 <div> 元素中：

示例

```
$("#div1").load("demo_test.txt");
```

也可以把 jQuery 选择器添加到 URL 参数。

下面的例子把 "demo_test.txt" 文件中 id="p1" 的元素的内容，加载到指定的 <div> 元素中：

实例

```
$("#div1").load("demo_test.txt #p1");
```

可选的 callback 参数规定当 load() 方法完成后所要允许的回调函数。回调函数可以设置不同的参数：

- *responseTxt* - 包含调用成功时的结果内容
- *statusTXT* - 包含调用的状态
- *xhr* - 包含 XMLHttpRequest 对象

下面的例子会在 load() 方法完成后显示一个提示框。如果 load() 方法已成功，则显示“外部内容加载成功！”，而如果失败，则显示错误消息：

实例

```
$("button").click(function(){
  $("#div1").load("demo_test.txt",function(responseTxt,statusTxt,xh
    if(statusTxt=="success")
      alert("外部内容加载成功！");
    if(statusTxt=="error")
      alert("Error: "+xhr.status+": "+xhr.statusText);
  });
});
```

jQuery AJAX 参考手册

如需完整的 AJAX 方法参考，请访问我们的 jQuery AJAX 参考手册。

jQuery - AJAX get() 和 post() 方法

jQuery get() 和 post() 方法用于通过 HTTP GET 或 POST 请求从服务器请求数据。

HTTP 请求：GET vs. POST

两种在客户端和服务器端进行请求-响应的常用方法是：GET 和 POST。

- *GET* - 从指定的资源请求数据
- *POST* - 向指定的资源提交要处理的数据

GET 基本上用于从服务器获得（取回）数据。注释：GET 方法可能返回缓存数据。

POST 也可用于从服务器获取数据。不过，POST 方法不会缓存数据，并且常用于连同请求一起发送数据。

如需学习更多有关 GET 和 POST 以及两方法差异的知识，请阅读我们的 HTTP 方法 - GET 对比 POST。

jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

```
$.get(_URL_,_callback_);
```

必需的 *URL* 参数规定您希望请求的 URL。

可选的 *callback* 参数是请求成功后所执行的函数名。

下面的例子使用 $.get() 方法从服务器上的一个文件中取回数据：

实例

```
$("button").click(function(){
  $.get("demo_test.asp",function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

$.get() 的第一个参数是我们希望请求的 URL（"demo_test.asp"）。

第二个参数是回调函数。第一个回调参数存有被请求页面的内容，第二个回调参数存有请求的状态。

提示：这个 ASP 文件 ("demo_test.asp") 类似这样：

```
<%
response.write("This is some text from an external ASP file.")
%>
```

jQuery $.post() 方法

$.post() 方法通过 HTTP POST 请求从服务器上请求数据。

语法：

```
$.post(_URL_,_data_,_callback_);
```

必需的 *URL* 参数规定您希望请求的 URL。

可选的 *data* 参数规定连同请求发送的数据。

可选的 *callback* 参数是请求成功后所执行的函数名。

下面的例子使用 $.post() 连同请求一起发送数据：

实例

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name:"Donald Duck",
    city:"Duckburg"
  },
  function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

$.post() 的第一个参数是我们希望请求的 URL ("demo_test_post.asp")。

然后我们连同请求（name 和 city）一起发送数据。

"demo_test_post.asp" 中的 ASP 脚本读取这些参数，对它们进行处理，然后返回结果。

第三个参数是回调函数。第一个回调参数存有被请求页面的内容，而第二个参数存有请求的状态。

提示：这个 ASP 文件 ("demo_test_post.asp") 类似这样：

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```

jQuery AJAX 参考手册

如需完整的 AJAX 方法参考，请访问我们的 jQuery AJAX 参考手册。

jQuery 杂项

jQuery - noConflict() 方法

如何在页面上同时使用 **jQuery** 和其他框架？

jQuery 和其他 JavaScript 框架

正如您已经了解到的，jQuery 使用 $ 符号作为 jQuery 的简写。

如果其他 JavaScript 框架也使用 $ 符号作为简写怎么办？

其他一些 JavaScript 框架包括：MooTools、Backbone、Sammy、Cappuccino、Knockout、JavaScript MVC、Google Web Toolkit、Google Closure、Ember、Batman 以及 Ext JS。

其中某些框架也使用 $ 符号作为简写（就像 jQuery），如果您在用的两种不同的框架正在使用相同的简写符号，有可能导致脚本停止运行。

jQuery 的团队考虑到了这个问题，并实现了 noConflict() 方法。

jQuery noConflict() 方法

noConflict() 方法会释放会 $ 标识符的控制，这样其他脚本就可以使用它了。

实例

当然，您仍然可以通过全名替代简写的方式来使用 jQuery：

```
$.noConflict();
jQuery(document).ready(function(){
  jQuery("button").click(function(){
    jQuery("p").text("jQuery 仍在运行！");
  });
});
```

实例

您也可以创建自己的简写。noConflict() 可返回对 jQuery 的引用，您可以把它存入变量，以供稍后使用。请看这个例子：

```
var jq = $.noConflict();
jq(document).ready(function(){
  jq("button").click(function(){
    jq("p").text("jQuery 仍在运行！");
  });
});
```

实例

如果你的 jQuery 代码块使用 $ 简写，并且您不愿意改变这个快捷方式，那么您可以把 $ 符号作为变量传递给 ready 方法。这样就可以在函数内使用 $ 符号了 - 而在函数外，依旧不得不使用 "jQuery"：

```
$.noConflict();
jQuery(document).ready(function($){
  $("button").click(function(){
    $("p").text("jQuery 仍在运行！");
  });
});
```

jQuery 核心参考手册

如需完整的 jQuery 核心方法概览，请访问我们的 jQuery 核心参考手册。

W3School jQuery 参考手册

来源：jQuery 参考手册

整理：飞龙

jQuery 参考手册 - 选择器

jQuery 选择器

选择器	实例	选取
*	$("*")	所有元素
#*id*	$("#lastname")	id="lastname" 的元素
.*class*	$(".intro")	所有 class="intro" 的元素
element	$("p")	所有 <p> 元素
.*class*.*class*	$(".intro.demo")	所有 class="intro" 且 class="demo" 的元素
:first	$("p:first")	第一个 <p> 元素
:last	$("p:last")	最后一个 <p> 元素
:even	$("tr:even")	所有偶数 <tr> 元素
:odd	$("tr:odd")	所有奇数 <tr> 元素
:eq(*index*)	$("ul li:eq(3)")	列表中的第四个元素（index 从 0 开始）
:gt(*no*)	$("ul li:gt(3)")	列出 index 大于 3 的元素
:lt(*no*)	$("ul li:lt(3)")	列出 index 小于 3 的元素
:not(*selector*)	$("input:not(:empty)")	所有不为空的 input 元素
:header	$(":header")	所有标题元素 <h1> - <h6>
:animated		所有动画元素
:contains(*text*)	$(":contains('W3School')")	包含指定字符串的所有元素
:empty	$(":empty")	无子（元素）节点的所有元素
:hidden	$("p:hidden")	所有隐藏的 <p> 元素
:visible	$("table:visible")	所有可见的表格
s1,s2,s3	$("th,td,.intro")	所有带有匹配选择的元素
[*attribute*]	$("[href]")	所有带有 href 属性的元素
		所有 href 属性的值等于 "#"

		的元素
[*attribute*!=*value*]	$("[href!='#']")	所有 href 属性的值不等于 "#" 的元素
[*attribute*$=*value*]	$("[href$='.jpg']")	所有 href 属性的值包含以 ".jpg" 结尾的元素
:input	$(":input")	所有 <input> 元素
:text	$(":text")	所有 type="text" 的 <input> 元素
:password	$(":password")	所有 type="password" 的 <input> 元素
:radio	$(":radio")	所有 type="radio" 的 <input> 元素
:checkbox	$(":checkbox")	所有 type="checkbox" 的 <input> 元素
:submit	$(":submit")	所有 type="submit" 的 <input> 元素
:reset	$(":reset")	所有 type="reset" 的 <input> 元素
:button	$(":button")	所有 type="button" 的 <input> 元素
:image	$(":image")	所有 type="image" 的 <input> 元素
:file	$(":file")	所有 type="file" 的 <input> 元素
:enabled	$(":enabled")	所有激活的 input 元素
:disabled	$(":disabled")	所有禁用的 input 元素
:selected	$(":selected")	所有被选取的 input 元素
:checked	$(":checked")	所有被选中的 input 元素

参阅

教程：jQuery 元素选择器语法

jQuery 参考手册 - 事件

jQuery 事件方法

事件方法会触发匹配元素的事件，或将函数绑定到所有匹配元素的某个事件。

触发实例：

```
$("button#demo").click()
```

上面的例子将触发 id="demo" 的 button 元素的 click 事件。

绑定实例：

```
$("button#demo").click(function(){$("img").hide()})
```

上面的例子会在点击 id="demo" 的按钮时隐藏所有图像。

方法	描述
bind()	向匹配元素附加一个或更多事件处理器
blur()	触发、或将函数绑定到指定元素的 blur 事件
change()	触发、或将函数绑定到指定元素的 change 事件
click()	触发、或将函数绑定到指定元素的 click 事件
dblclick()	触发、或将函数绑定到指定元素的 double click 事件
delegate()	向匹配元素的当前或未来的子元素附加一个或多个事件处理器
die()	移除所有通过 live() 函数添加的事件处理程序。
error()	触发、或将函数绑定到指定元素的 error 事件
event.isDefaultPrevented()	返回 event 对象上是否调用了 event.preventDefault()。
event.pageX	相对于文档左边缘的鼠标位置。
event.pageY	相对于文档上边缘的鼠标位置。
event.preventDefault()	阻止事件的默认动作。
	包含由被指定事件触发的事件处理器返回的最后

	一个值。
event.target	触发该事件的 DOM 元素。
event.timeStamp	该属性返回从 1970 年 1 月 1 日到事件发生时的毫秒数。
event.type	描述事件的类型。
event.which	指示按了哪个键或按钮。
focus()	触发、或将函数绑定到指定元素的 focus 事件
keydown()	触发、或将函数绑定到指定元素的 key down 事件
keypress()	触发、或将函数绑定到指定元素的 key press 事件
keyup()	触发、或将函数绑定到指定元素的 key up 事件
live()	为当前或未来的匹配元素添加一个或多个事件处理器
load()	触发、或将函数绑定到指定元素的 load 事件
mousedown()	触发、或将函数绑定到指定元素的 mouse down 事件
mouseenter()	触发、或将函数绑定到指定元素的 mouse enter 事件
mouseleave()	触发、或将函数绑定到指定元素的 mouse leave 事件
mousemove()	触发、或将函数绑定到指定元素的 mouse move 事件
mouseout()	触发、或将函数绑定到指定元素的 mouse out 事件
mouseover()	触发、或将函数绑定到指定元素的 mouse over 事件
mouseup()	触发、或将函数绑定到指定元素的 mouse up 事件
one()	向匹配元素添加事件处理器。每个元素只能触发一次该处理器。
ready()	文档就绪事件（当 HTML 文档就绪可用时）
resize()	触发、或将函数绑定到指定元素的 resize 事件
scroll()	触发、或将函数绑定到指定元素的 scroll 事件

select()	触发、或将函数绑定到指定元素的 select 事件
submit()	触发、或将函数绑定到指定元素的 submit 事件
toggle()	绑定两个或多个事件处理器函数，当发生轮流的 click 事件时执行。
trigger()	所有匹配元素的指定事件
triggerHandler()	第一个被匹配元素的指定事件
unbind()	从匹配元素移除一个被添加的事件处理器
undelegate()	从匹配元素移除一个被添加的事件处理器，现在或将来
unload()	触发、或将函数绑定到指定元素的 unload 事件

参阅

教程：jQuery 元素选择器语法

jQuery 事件 - bind() 方法

实例

当点击鼠标时，隐藏或显示 p 元素：

```
$("button").bind("click",function(){
  $("p").slideToggle();
});
```

定义和用法

bind() 方法为被选元素添加一个或多个事件处理程序，并规定事件发生时运行的函数。

将事件和函数绑定到元素

规定向被选元素添加的一个或多个事件处理程序，以及当事件发生时运行的函数。

语法

```
$(_selector_).bind(_event_,_data_,_function_)
```

参数	描述
event	必需。规定添加到元素的一个或多个事件。由空格分隔多个事件。必须是有效的事件。
data	可选。规定传递到函数的额外数据。
function	必需。规定当事件发生时运行的函数。

替代语法

```
$(_selector_).bind({_event_:_function_, _event_:_function_, ...})
```

参数	描述
{*event*:*function*, *event*:*function*, ...}	必需。规定事件映射，其中包含一个或多个添加到元素的事件，以及当事件发生时运行的函数。

jQuery 事件 - blur() 方法

实例

当输入域失去焦点 (blur) 时改变其颜色：

```
$("input").blur(function(){
  $("input").css("background-color","#D6D6FF");
});
```

定义和用法

当元素失去焦点时发生 blur 事件。

blur() 函数触发 blur 事件，或者如果设置了 *function* 参数，该函数也可规定当发生 blur 事件时执行的代码。

提示：早前，blur 事件仅发生于表单元素上。在新浏览器中，该事件可用于任何元素。

触发 **blur** 事件

触发被选元素的 blur 事件。

语法

```
$(_selector_).blur()
```

将函数绑定到 **blur** 事件

规定当被选元素的 blur 事件发生时运行的函数。

语法

```
$(_selector_).blur(_function_)
```

参数	描述
function	可选。规定当 blur 事件发生时运行的函数。

jQuery 事件 - change() 方法

实例

当输入域发生变化时改变其颜色：

```
$(".field").change(function(){
  $(this).css("background-color","#FFFFCC");
});
```

定义和用法

当元素的值发生改变时，会发生 change 事件。

该事件仅适用于文本域（text field），以及 textarea 和 select 元素。

change() 函数触发 change 事件，或规定当发生 change 事件时运行的函数。

注释：当用于 select 元素时，change 事件会在选择某个选项时发生。当用于 text field 或 text area 时，该事件会在元素失去焦点时发生。

触发 change 事件

触发被选元素的 change 事件。

语法

```
$(_selector_).change()
```

将函数绑定到 change 事件

规定当被选元素的 change 事件发生时运行的函数。

语法

```
$(_selector_).change(_function_)
```

参数	描述
function	可选。规定当 change 事件发生时运行的函数。

jQuery 事件 - click() 方法

实例

当点击按钮时，隐藏或显示元素：

```
$("button").click(function(){
  $("p").slideToggle();
});
```

定义和用法

当点击元素时，会发生 click 事件。

当鼠标指针停留在元素上方，然后按下并松开鼠标左键时，就会发生一次 click。

click() 方法触发 click 事件，或规定当发生 click 事件时运行的函数。

触发 click 事件

语法

```
$(_selector_).click()
```

将函数绑定到 click 事件

语法

```
$(_selector_).click(_function_)
```

参数	描述
function	可选。规定当发生 click 事件时运行的函数。

jQuery 事件 - dblclick() 方法

实例

当双击按钮时，隐藏或显示元素：

```
$("button").dblclick(function(){
  $("p").slideToggle();
});
```

定义和用法

当双击元素时，会发生 dblclick 事件。

当鼠标指针停留在元素上方，然后按下并松开鼠标左键时，就会发生一次 click。

在很短的时间内发生两次 click，即是一次 double click 事件。

dblclick() 方法触发 dblclick 事件，或规定当发生 dblclick 事件时运行的函数。

提示：如果把 dblclick 和 click 事件应用于同一元素，可能会产生问题。

触发 dblclick 事件

语法

```
$(_selector_).dblclick()
```

将函数绑定到 dblclick 事件

语法

```
$(_selector_).dblclick(_function_)
```

参数	描述
function	可选。规定当发生 dblclick 事件时运行的函数。

jQuery 事件 - delegate() 方法

实例

当点击鼠标时，隐藏或显示 p 元素：

```
$("div").delegate("button","click",function(){
  $("p").slideToggle();
});
```

定义和用法

delegate() 方法为指定的元素（属于被选元素的子元素）添加一个或多个事件处理程序，并规定当这些事件发生时运行的函数。

使用 delegate() 方法的事件处理程序适用于当前或未来的元素（比如由脚本创建的新元素）。

语法

```
$(_selector_).delegate(_childSelector_,_event_,_data_,_function_)
```

参数	描述
childSelector	必需。规定要附加事件处理程序的一个或多个子元素。
event	必需。规定附加到元素的一个或多个事件。由空格分隔多个事件值。必须是有效的事件。
data	可选。规定传递到函数的额外数据。
function	必需。规定当事件发生时运行的函数。

亲自试一试 - 实例

向未来的元素添加事件处理程序

如何使用 delegate() 方法向尚未创建的元素添加事件处理程序。

jQuery 事件 - die() 方法

实例

移除所有通过 live() 方法向 p 元素添加的事件处理程序：

```
$("p").die();
```

定义和用法

die() 方法移除所有通过 live() 方法向指定元素添加的一个或多个事件处理程序。

语法

```
$(_selector_).die(_event_,_function_)
```

参数	描述
event	必需。规定要移除的一个或多个事件处理程序。由空格分隔多个事件值。必须是有效的事件。
function	可选。规定要移除的特定函数。

亲自试一试 - 实例

移除通过 live() 添加的特定函数

如何使用 live() 方法移除事件处理程序的特定函数。

jQuery 事件 - error() 方法

实例

如果图像不存在，则用一段预定义的文本取代它：

```
$("img").error(function(){
  $("img").replaceWith("

Missing image!

");
});
```

定义和用法

当元素遇到错误（没有正确载入）时，发生 error 事件。

error() 方法触发 error 事件，或规定当发生 error 事件时运行的函数。

提示：该方法是 bind('error', handler) 的简写方式。

触发 error 事件

语法

```
$(_selector_).error()
```

将函数绑定到 error 事件

语法

```
$(_selector_).error(_function_)
```

参数	描述
function	可选。规定当发生 error 事件时运行的函数。

jQuery 事件 - isDefaultPrevented() 方法

实例

防止链接打开 URL，并声明来自 isDefaultPrevented() 的结果：

```
$("a").click(function(event){
   event.preventDefault();
  alert("Default prevented: " + event.isDefaultPrevented());
});
```

定义和用法

isDefaultPrevented() 方法返回指定的 event 对象上是否调用了 preventDefault() 方法。

语法

```
_event_.isDefaultPrevented()
```

参数	描述
event	必需。规定需要检查的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - pageX 属性

实例

显示鼠标指针的位置：

```
$(document).mousemove(function(e){
  $("span").text("X: " + e.pageX + ", Y: " + e.pageY);
});
```

定义和用法

pageX() 属性是鼠标指针的位置，相对于文档的左边缘。

语法

```
_event_.pageX
```

参数	描述
event	必需。规定要使用的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - pageY 属性

实例

显示鼠标指针的位置：

```
$(document).mousemove(function(e){
  $("span").text("X: " + e.pageX + ", Y: " + e.pageY);
});
```

定义和用法

pageY() 属性是鼠标指针的位置，相对于文档的上边缘。

语法

```
_event_.pageY
```

参数	描述
event	必需。规定要使用的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - preventDefault() 方法

实例

防止链接打开 URL：

```
$("a").click(function(event){
  event.preventDefault();
});
```

定义和用法

preventDefault() 方法阻止元素发生默认的行为（例如，当点击提交按钮时阻止对表单的提交）。

语法

```
_event_.preventDefault()
```

参数	描述
event	必需。规定阻止哪个事件的默认动作。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - result 属性

实例

显示最后一次点击事件返回的结果：

```
$("button").click(function(e) {
  $("p").html(e.result);
});
```

定义和用法

result 属性包含由被指定事件触发的事件处理器返回的最后一个值，除非这个值未定义。

语法

```
_event_.result
```

参数	描述
event	必需。规定返回的最后一个值来自哪个事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - target 属性

实例

显示哪个 DOM 元素触发了事件：

```
$("p, button, h1, h2").click(function(event){
  $("div").html("Triggered by a " + event.target.nodeName + " eleme
});
```

定义和用法

target 属性规定哪个 DOM 元素触发了该事件。

语法

```
_event_.target
```

参数	描述
event	必需。规定需要检查的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - timeStamp 属性

实例

显示出当对按钮元素的点击事件发生时的时间戳：

```
$("button").click(function(event){
  $("span")html(event.timeStamp);
});
```

定义和用法

timeStamp 属性包含从 1970 年 1 月 1 日到事件被触发时的毫秒数。

语法

```
_event_.timeStamp
```

参数	描述
event	必需。规定返回该时间戳的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - type 属性

实例

显示触发了哪种类型的事件：

```
$("p").bind('click dblclick mouseover mouseout',function(event){
  $("div").html("Event: " + event.type);
});
```

定义和用法

type 属性描述触发哪种事件类型。

语法

```
_event_.type
```

参数	描述
event	必需。规定要检查的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - which 属性

实例

显示按了哪个键：

```
$("input").keydown(function(event){
  $("div").html("Key: " + event.which);
});
```

定义和用法

which 属性指示按了哪个键或按钮。

语法

```
_event_.which
```

参数	描述
event	必需。规定要检查的事件。这个 *event* 参数来自事件绑定函数。

jQuery 事件 - focus() 方法

实例

当输入框获得焦点时，改变它的背景色：

```
$("input").focus(function(){
  $("input").css("background-color","#FFFFCC");
});
```

定义和用法

当元素获得焦点时，发生 focus 事件。

当通过鼠标点击选中元素或通过 tab 键定位到元素时，该元素就会获得焦点。

focus() 方法触发 focus 事件，或规定当发生 focus 事件时运行的函数。

触发 **focus** 事件

语法

```
$(_selector_).focus()
```

将函数绑定到 **focus** 事件

语法

```
$(_selector_).focus(_function_)
```

参数	描述
function	可选。规定当发生 focus 事件时运行的函数。

jQuery 事件 - keydown() 方法

实例

当按下按键时，改变文本域的颜色：

```
$("input").keydown(function(){
  $("input").css("background-color","#FFFFCC");
});
```

定义和用法

完整的 key press 过程分为两个部分：1. 按键被按下；2. 按键被松开。

当按钮被按下时，发生 keydown 事件。

keydown() 方法触发 keydown 事件，或规定当发生 keydown 事件时运行的函数。

注释：如果在文档元素上进行设置，则无论元素是否获得焦点，该事件都会发生。

提示：请使用 .which 属性来确定按下了哪个按键（亲自试一试）。

触发 keydown 事件

语法

```
$(_selector_).keydown()
```

将函数绑定到 keydown 事件

语法

```
$(_selector_).keydown(_function_)
```

参数	描述
function	可选。规定当发生 keydown 事件时运行的函数。

jQuery 事件 - keypress() 方法

实例

计算在输入域中的按键次数：

```
$("input").keydown(function(){
  $("span").text(i+=1);
});
```

定义和用法

keypress 事件与 keydown 事件类似。当按钮被按下时，会发生该事件。它发生在当前获得焦点的元素上。

不过，与 keydown 事件不同，每插入一个字符，就会发生 keypress 事件。

keypress() 方法触发 keypress 事件，或规定当发生 keypress 事件时运行的函数。

注释：如果在文档元素上进行设置，则无论元素是否获得焦点，该事件都会发生。

触发 **keypress** 事件

语法

```
$(_selector_).keypress()
```

将函数绑定到 **keypress** 事件

语法

```
$(_selector_).keypress(_function_)
```

参数	描述
function	可选。规定当发生 keypress 事件时运行的函数。

jQuery 事件 - keyup() 方法

实例

当按下按键时，改变文本域的颜色：

```
$("input").keyup(function(){
  $("input").css("background-color","#D6D6FF");
});
```

定义和用法

完整的 key press 过程分为两个部分，按键被按下，然后按键被松开并复位。

当按钮被松开时，发生 keyup 事件。它发生在当前获得焦点的元素上。

keyup() 方法触发 keyup 事件，或规定当发生 keyup 事件时运行的函数。

注释：如果在文档元素上进行设置，则无论元素是否获得焦点，该事件都会发生。

提示：请使用 .which 属性来确定按下了哪个按键（亲自试一试）。

触发 keyup 事件

语法

```
$(_selector_).keyup()
```

将函数绑定到 keyup 事件

语法

```
$(_selector_).keyup(_function_)
```

参数	描述
function	可选。规定当发生 keyup 事件时运行的函数。

jQuery 事件 - live() 方法

实例

当点击按钮时，隐藏或显示 p 元素：

```
$("button").live("click",function(){
  $("p").slideToggle();
});
```

定义和用法

live() 方法为被选元素附加一个或多个事件处理程序，并规定当这些事件发生时运行的函数。

通过 live() 方法附加的事件处理程序适用于匹配选择器的当前及未来的元素（比如由脚本创建的新元素）。

语法

```
$(_selector_).live(_event_,_data_,_function_)
```

参数	描述
event	必需。规定附加到元素的一个或多个事件。由空格分隔多个事件。必须是有效的事件。
data	可选。规定传递到该函数的额外数据。
function	必需。规定当事件发生时运行的函数。

亲自试一试 - 实例

向未来的元素添加事件处理器

如何使用 live() 方法向尚未创建的元素添加事件处理器。

jQuery 事件 - load() 方法

实例

当图像加载时，改变 div 元素的文本：

```
$("img").load(function(){
  $("div").text("Image loaded");
});
```

定义和用法

当指定的元素（及子元素）已加载时，会发生 load() 事件。

该事件适用于任何带有 URL 的元素（比如图像、脚本、框架、内联框架）。

根据不同的浏览器（Firefox 和 IE），如果图像已被缓存，则也许不会触发 load 事件。

注释：还存在一个名为 load() 的 jQuery Ajax 方法，根据不同的参数而定。

语法

```
$(_selector_).load(_function_)
```

参数	描述
function	必需。规定当指定元素加载完成时运行的函数。

jQuery 事件 - mousedown() 方法

实例

当按下鼠标按钮时，隐藏或显示元素：

```
$("button").mousedown(function(){
  $("p").slideToggle();
});
```

定义和用法

当鼠标指针移动到元素上方，并按下鼠标按键时，会发生 mousedown 事件。

与 click 事件不同，mousedown 事件仅需要按键被按下，而不需要松开即可发生。

mousedown() 方法触发 mousedown 事件，或规定当发生 mousedown 事件时运行的函数。

触发 **mousedown** 事件

语法

```
$(_selector_).mousedown()
```

将函数绑定到 **mousedown** 事件

语法

```
$(_selector_).mousedown(_function_)
```

参数	描述
function	可选。规定当发生 mousedown 事件时运行的函数。

jQuery 事件 - mouseenter() 方法

实例

当鼠标指针进入（穿过）元素时，改变元素的背景色：

```
$("p").mouseenter(function(){
  $("p").css("background-color","yellow");
});
```

定义和用法

当鼠标指针穿过元素时，会发生 mouseenter 事件。

该事件大多数时候会与 mouseleave 事件一起使用。

mouseenter() 方法触发 mouseenter 事件，或规定当发生 mouseenter 事件时运行的函数。

注释：与 mouseover 事件不同，只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件。如果鼠标指针穿过任何子元素，同样会触发 mouseover 事件。请看下面例子的演示。

亲自试一试：mouseenter 与 mouseover 的不同

触发 mouseenter 事件

语法

```
$(_selector_).mouseenter()
```

将函数绑定到 mouseenter 事件

语法

```
$(_selector_).mouseenter(_function_)
```

参数	描述
function	可选。规定当发生 mouseenter 事件时运行的函数。

jQuery 事件 - mouseleave() 方法

实例

当鼠标指针离开元素时，改变元素的背景色：

```
$("p").mouseleave(function(){
  $("p").css("background-color","#E9E9E4");
});
```

定义和用法

当鼠标指针离开元素时，会发生 mouseleave 事件。

该事件大多数时候会与 mouseenter 事件一起使用。

mouseleave() 方法触发 mouseleave 事件，或规定当发生 mouseleave 事件时运行的函数。

注释：与 mouseout 事件不同，只有在鼠标指针离开被选元素时，才会触发 mouseleave 事件。如果鼠标指针离开任何子元素，同样会触发 mouseout 事件。请看下面例子的演示。

亲自试一试：mouseleave 与 mouseout 的不同

触发 mouseleave 事件

语法

```
$(_selector_).mouseleave()
```

将函数绑定到 mouseleave 事件

语法

```
$(_selector_).mouseleave(_function_)
```

参数	描述
function	可选。规定当发生 mouseleave 事件时运行的函数。

jQuery 事件 - mousemove() 方法

实例

获得鼠标指针在页面中的位置：

```
$(document).mousemove(function(e){
  $("span").text(e.pageX + ", " + e.pageY);
});
```

定义和用法

当鼠标指针在指定的元素中移动时，就会发生 mousemove 事件。

mousemove() 方法触发 mousemove 事件，或规定当发生 mousemove 事件时运行的函数。

注意：用户把鼠标移动一个像素，就会发生一次 mousemove 事件。处理所有 mousemove 事件会耗费系统资源。请谨慎使用该事件。

触发 mousemove 事件

语法

```
$(_selector_).mousemove()
```

将函数绑定到 mousemove 事件

语法

```
$(_selector_).mousemove(_function_)
```

参数	描述
function	可选。规定当发生 mousemove 事件时运行的函数。

jQuery 事件 - mouseout() 方法

实例

当鼠标从元素上移开时，改变元素的背景色：

```
$("p").mouseout(function(){
  $("p").css("background-color","#E9E9E4");
});
```

定义和用法

当鼠标指针从元素上移开时，发生 mouseout 事件。

该事件大多数时候会与 mouseover 事件一起使用。

mouseout() 方法触发 mouseout 事件，或规定当发生 mouseout 事件时运行的函数。

注释：与 mouseleave 事件不同，不论鼠标指针离开被选元素还是任何子元素，都会触发 mouseout 事件。只有在鼠标指针离开被选元素时，才会触发 mouseleave 事件。请看下面例子的演示。

亲自试一试：mouseleave 与 mouseout 的不同

触发 mouseout 事件

语法

```
$(_selector_).mouseout()
```

将函数绑定到 mouseout 事件

语法

```
$(_selector_).mouseout(_function_)
```

参数	描述
function	可选。规定当发生 mouseout 事件时运行的函数。

jQuery 事件 - mouseover() 方法

实例

当鼠标指针位于元素上方时时，改变元素的背景色：

```
$("p").mouseover(function(){
  $("p").css("background-color","yellow");
});
```

定义和用法

当鼠标指针位于元素上方时，会发生 mouseover 事件。

该事件大多数时候会与 mouseout 事件一起使用。

mouseover() 方法触发 mouseover 事件，或规定当发生 mouseover 事件时运行的函数。

注释：与 mouseenter 事件不同，不论鼠标指针穿过被选元素或其子元素，都会触发 mouseover 事件。只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件。请看下面例子的演示。

亲自试一试：mouseenter 与 mouseover 的不同

触发 mouseover 事件

语法

```
$(_selector_).mouseover()
```

将函数绑定到 mouseover 事件

语法

```
$(_selector_).mouseover(_function_)
```

参数	描述
function	可选。规定当发生 mouseover 事件时运行的函数。

jQuery 事件 - mouseup() 方法

实例

当松开鼠标按钮时，隐藏或显示元素：

```
$("button").mouseup(function(){
  $("p").slideToggle();
});
```

定义和用法

当在元素上放松鼠标按钮时，会发生 mouseup 事件。

与 click 事件不同，mouseup 事件仅需要放松按钮。当鼠标指针位于元素上方时，放松鼠标按钮就会触发该事件。

mouseup() 方法触发 mouseup 事件，或规定当发生 mouseup 事件时运行的函数。

触发 mouseup 事件

语法

```
$(_selector_).mouseup()
```

将函数绑定到 mouseup 事件

语法

```
$(_selector_).mouseup(_function_)
```

参数	描述
function	可选。规定当发生 mouseup 事件时运行的函数。

jQuery 事件 - one() 方法

实例

当点击 p 元素时，增加该元素的文本大小：

```
$("p").one("click",function(){
  $(this).animate({fontSize:"+=6px"});
});
```

定义和用法

one() 方法为被选元素附加一个或多个事件处理程序，并规定当事件发生时运行的函数。

当使用 one() 方法时，每个元素只能运行一次事件处理器函数。

语法

```
$(_selector_).one(_event_,_data_,_function_)
```

参数	描述
event	必需。规定添加到元素的一个或多个事件。由空格分隔多个事件。必须是有效的事件。
data	可选。规定传递到函数的额外数据。
function	必需。规定当事件发生时运行的函数。

jQuery 事件 - ready() 方法

实例

在文档加载后激活函数：

```
$(document).ready(function(){
  $(".btn1").click(function(){
    $("p").slideToggle();
  });
});
```

定义和用法

当 DOM（文档对象模型） 已经加载，并且页面（包括图像）已经完全呈现时，会发生 ready 事件。

由于该事件在文档就绪后发生，因此把所有其他的 jQuery 事件和函数置于该事件中是非常好的做法。正如上面的例子中那样。

ready() 函数规定当 ready 事件发生时执行的代码。

ready() 函数仅能用于当前文档，因此无需选择器。

允许使用以下三种语法：

语法 1

```
$(document).ready(_function_)
```

语法 2

```
$().ready(_function_)
```

语法 3

```
$(_function_)
```

参数	描述
function	必需。规定当文档加载后要运行的函数。

提示和注释

提示：ready() 函数不应与 <body onload=""> 一起使用。

jQuery 事件 - resize() 方法

实例

对浏览器窗口调整大小进行计数：

```
$(window).resize(function() {
  $('span').text(x+=1);
});
```

定义和用法

当调整浏览器窗口的大小时，发生 resize 事件。

resize() 方法触发 resize 事件，或规定当发生 resize 事件时运行的函数。

触发 **resize** 事件

语法

```
$(_selector_).resize()
```

将函数绑定到 **resize** 事件

语法

```
$(_selector_).resize(_function_)
```

参数	描述
function	可选。规定当发生 resize 事件时运行的函数。

jQuery 事件 - scroll() 方法

实例

对元素滚动的次数进行计数：

```
$("div").scroll(function() {
  $("span").text(x+=1);
});
```

定义和用法

当用户滚动指定的元素时，会发生 scroll 事件。

scroll 事件适用于所有可滚动的元素和 window 对象（浏览器窗口）。

scroll() 方法触发 scroll 事件，或规定当发生 scroll 事件时运行的函数。

触发 **scroll** 事件

语法

```
$(_selector_).scroll()
```

将函数绑定到 **scroll** 事件

语法

```
$(_selector_).scroll(_function_)
```

参数	描述
function	可选。规定当发生 scroll 事件时运行的函数。

jQuery 事件 - select() 方法

实例

在文本域后添加文本，以显示出提示文本：

```
$("input").select(function(){
  $("input").after(" Text marked!");
});
```

定义和用法

当 textarea 或文本类型的 input 元素中的文本被选择时，会发生 select 事件。

select() 方法触发 select 事件，或规定当发生 select 事件时运行的函数。

触发 **select** 事件

语法

```
$(_selector_).select()
```

将函数绑定到 **select** 事件

语法

```
$(_selector_).select(_function_)
```

参数	描述
function	可选。规定当发生 select 事件时运行的函数。

jQuery 事件 - submit() 方法

实例

当提交表单时，显示警告框：

```
$("form").submit(function(e){
  alert("Submitted");
});
```

定义和用法

当提交表单时，会发生 submit 事件。

该事件只适用于表单元素。

submit() 方法触发 submit 事件，或规定当发生 submit 事件时运行的函数。

触发 submit 事件

语法

```
$(_selector_).submit()
```

将函数绑定到 submit 事件

语法

```
$(_selector_).submit(_function_)
```

参数	描述
function	可选。规定当发生 submit 事件时运行的函数。

亲自试一试 - 实例

阻止提交按钮的默认 action

使用 preventDefault() 函数来阻止对表单的提交。

jQuery 事件 - toggle() 方法

实例

切换不同的背景色：

```
$("p").toggle(
  function(){
  $("body").css("background-color","green");},
  function(){
  $("body").css("background-color","red");},
  function(){
  $("body").css("background-color","yellow");}
);
```

定义和用法

toggle() 方法用于绑定两个或多个事件处理器函数，以响应被选元素的轮流的 click 事件。

该方法也可用于切换被选元素的 hide() 与 show() 方法。

向 Toggle 事件绑定两个或更多函数

当指定元素被点击时，在两个或多个函数之间轮流切换。

如果规定了两个以上的函数，则 toggle() 方法将切换所有函数。例如，如果存在三个函数，则第一次点击将调用第一个函数，第二次点击调用第二个函数，第三次点击调用第三个函数。第四次点击再次调用第一个函数，以此类推。

语法

```
$(_selector_).toggle(_function1()_,_function2()_,_functionN()_,...
```

参数	描述
function1()	必需。规定当元素在每偶数次被点击时要运行的函数。
function2()	必需。规定当元素在每奇数次被点击时要运行的函数。
functionN(),...	可选。规定需要切换的其他函数。

切换 Hide() 和 Show()

检查每个元素是否可见。

如果元素已隐藏，则运行 show()。如果元素可见，则元素 hide()。这样就可以创造切换效果。

语法

```
$(_selector_).toggle(_speed_,_callback_)
```

参数	描述
speed	可选。规定 hide/show 效果的速度。默认是 "0"。可能的值： 毫秒（比如 1500） "slow" "normal" "fast"
callback	可选。当 toggle() 方法完成时执行的函数。如需学习更多有关 callback 的知识，请访问我们的 Callback 函数教程。

显示或隐藏元素

规定是否只显示或只隐藏所有匹配的元素。

语法

```
$(_selector_).toggle(_switch_)
```

参数	描述
switch	必需。布尔值，规定 toggle() 是否应只显示或只隐藏所有被选元素。 `true` - 显示元素 `false` - 隐藏元素

jQuery 事件 - trigger() 方法

实例

触发 input 元素的 select 事件：

```
$("button").click(function(){
  $("input").trigger("select");
});
```

定义和用法

trigger() 方法触发被选元素的指定事件类型。

触发事件

规定被选元素要触发的事件。

语法

```
$(_selector_).trigger(_event_,[_param1_,_param2_,...])
```

参数	描述
event	必需。规定指定元素要触发的事件。可以使自定义事件（使用 bind() 函数来附加），或者任何标准事件。
[*param1*,*param2*,...]	可选。传递到事件处理程序的额外参数。额外的参数对自定义事件特别有用。

使用 Event 对象来触发事件

规定使用事件对象的被选元素要触发的事件。

语法

```
$(_selector_).trigger(_eventObj_)
```

参数	描述
eventObj	必需。规定事件发生时运行的函数。

jQuery 事件 - triggerHandler() 方法

实例

触发 input 元素的 select 事件：

```
$("button").click(function(){
  $("input").triggerHandler("select");
});
```

定义和用法

triggerHandler() 方法触发被选元素的指定事件类型。但不会执行浏览器默认动作，也不会产生事件冒泡。

triggerHandler() 方法与 trigger() 方法类似。不同的是它不会触发事件（比如表单提交）的默认行为，而且只影响第一个匹配元素。

与 **trigger()** 方法相比的不同之处

- 它不会引起事件（比如表单提交）的默认行为
- .trigger() 会操作 jQuery 对象匹配的所有元素，而 .triggerHandler() 只影响第一个匹配元素。
- 由 .triggerHandler() 创建的事件不会在 DOM 树中冒泡；如果目标元素不直接处理它们，则不会发生任何事情。
- 该方法的返回的是事件处理函数的返回值，而不是具有可链性的 jQuery 对象。此外，如果没有处理程序被触发，则这个方法返回 undefined。

触发事件

规定被选元素要触发的事件。

语法

```
$(_selector_).triggerHandler(_event_,[_param1_,_param2_,...])
```

参数	描述
event	必需。规定指定元素要触发的事件。
[*param1*,*param2*,...]	可选。传递到事件处理程序的额外参数。

jQuery 事件 - unbind() 方法

实例

移除所有 p 元素的事件处理器：

```
$("button").click(function(){
  $("p").unbind();
});
```

定义和用法

unbind() 方法移除被选元素的事件处理程序。

该方法能够移除所有的或被选的事件处理程序，或者当事件发生时终止指定函数的运行。

ubind() 适用于任何通过 jQuery 附加的事件处理程序。

取消绑定元素的事件处理程序和函数

规定从指定元素上删除的一个或多个事件处理程序。

如果没有规定参数，unbind() 方法会删除指定元素的所有事件处理程序。

语法

```
$(_selector_).unbind(_event_,_function_)
```

参数	描述
event	可选。规定删除元素的一个或多个事件由空格分隔多个事件值。如果只规定了该参数，则会删除绑定到指定事件的所有函数。
function	可选。规定从元素的指定事件取消绑定的函数名。

使用 Event 对象来取消绑定事件处理程序

规定要删除的事件对象。用于对自身内部的事件取消绑定（比如当事件已被触发一定次数之后，删除事件处理程序）。

如果未规定参数，则 unbind() 方法会删除指定元素的所有事件处理程序。

语法

```
$(_selector_).unbind(_eventObj_)
```

参数	描述
eventObj	可选。规定要使用的事件对象。这个 eventObj 参数来自事件绑定函数。

亲自试一试 - 实例

取消绑定特定的函数

如何使用 unbind() 方法取消绑定元素指定事件的具体函数。

jQuery 事件 - undelegate() 方法

实例

从所有元素删除由 delegate() 方法添加的所有事件处理器：

```
$("body").undelegate();
```

定义和用法

undelegate() 方法删除由 delegate() 方法添加的一个或多个事件处理程序。

语法

```
$(_selector_).undelegate(_selector_,_event_,_function_)
```

参数	描述
selector	可选。规定需要删除事件处理程序的选择器。
event	可选。规定需要删除处理函数的一个或多个事件类型。
function	可选。规定要删除的具体事件处理函数。

亲自试一试 - 实例

删除事件处理程序，由 delegate() 添加，来自具体选择器

如何使用 undelegate() 方法从指定元素删除所有事件处理程序。

删除指定事件类型的事件处理程序，由 delegate() 添加，来自指定元素

如何使用 undelegate() 方法从指定元素删除具体事件类型的所有事件处理程序。

删除由 delegate() 添加的具体函数

如何使用 undelegate() 方法为事件处理程序删除来自特定事件类型的特定函数。

jQuery 事件 - unload 属性

实例

当用户点击链接离开本页时，弹出一个消息框：

```
$(window).unload(function(){
  alert("Goodbye!");
});
```

定义和用法

当用户离开页面时，会发生 unload 事件。

具体来说，当发生以下情况时，会发出 unload 事件：

- 点击某个离开页面的链接
- 在地址栏中键入了新的 URL
- 使用前进或后退按钮
- 关闭浏览器
- 重新加载页面

unload() 方法将事件处理程序绑定到 unload 事件。

unload() 方法只应用于 window 对象。

语法

```
_event_.unload(_function_)
```

参数	描述
function	必需。规定当触发 unload 事件时运行的函数。

jQuery 参考手册 - 效果

jQuery 效果函数

方法	描述
animate()	对被选元素应用“自定义”的动画
clearQueue()	对被选元素移除所有排队的函数（仍未运行的）
delay()	对被选元素的所有排队函数（仍未运行）设置延迟
dequeue()	运行被选元素的下一个排队函数
fadeIn()	逐渐改变被选元素的不透明度，从隐藏到可见
fadeOut()	逐渐改变被选元素的不透明度，从可见到隐藏
fadeTo()	把被选元素逐渐改变至给定的不透明度
hide()	隐藏被选的元素
queue()	显示被选元素的排队函数
show()	显示被选的元素
slideDown()	通过调整高度来滑动显示被选元素
slideToggle()	对被选元素进行滑动隐藏和滑动显示的切换
slideUp()	通过调整高度来滑动隐藏被选元素
stop()	停止在被选元素上运行动画
toggle()	对被选元素进行隐藏和显示的切换

jQuery 效果 - animate() 方法

实例

改变 "div" 元素的高度：

```
$(".btn1").click(function(){
  $("#box").animate({height:"300px"});
});
```

定义和用法

animate() 方法执行 CSS 属性集的自定义动画。

该方法通过CSS样式将元素从一个状态改变为另一个状态。CSS属性值是逐渐改变的，这样就可以创建动画效果。

只有数字值可创建动画（比如 "margin:30px"）。字符串值无法创建动画（比如 "background-color:red"）。

注释：使用 "+=" 或 "-=" 来创建相对动画（relative animations）。

语法 1

```
$(_selector_).animate(_styles_,_speed_,_easing_,_callback_)
```

参数	描述
styles	必需。规定产生动画效果的 CSS 样式和值。可能的 CSS 样式值（提供实例）： `backgroundPosition` `borderWidth` `borderBottomWidth` `borderLeftWidth` `borderRightWidth` `borderTopWidth` `borderSpacing` `margin` `marginBottom` `marginLeft` `marginRight` `marginTop` `outlineWidth` `padding` `paddingBottom` `paddingLeft` `paddingRight` `paddingTop` `height` `width` `maxHeight` `maxWidth` `minHeight` `minWidth` `font` `fontSize` `bottom` `left` `right` `top` `letterSpacing` `wordSpacing` `lineHeight` `textIndent` 注释：CSS 样式使用 DOM 名称（比如 "fontSize"）来设置，而非 CSS 名称（比如 "font-size"）。
speed	可选。规定动画的速度。默认是 "normal"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"
easing	可选。规定在不同的动画点中设置动画速度的 easing 函数。内置的 easing 函数： `swing` `linear` 扩展插件中提供更多 easing 函数。
callback	可选。animate 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。

语法 2

```
$(_selector_).animate(_styles_,_options_)
```

参数	描述
styles	必需。规定产生动画效果的 CSS 样式和值（同上）。
options	可选。规定动画的额外选项。可能的值： `speed` - 设置动画的速度 `easing` - 规定要使用的 easing 函数 `callback` - 规定动画完成之后要执行的函数 `step` - 规定动画的每一步完成之后要执行的函数 `queue` - 布尔值。指示是否在效果队列中放置动画。如果为 false，则动画将立即开始 `specialEasing` - 来自 *styles* 参数的一个或多个 CSS 属性的映射，以及它们的对应 easing 函数

jQuery 效果 - clearQueue() 方法

实例

停止当前正在运行的动画：

```
$("#stop").click(function(){
  $("#box").clearQueue();
});
```

定义和用法

clearQueue() 方法停止队列中所有仍未执行的函数。

与 stop() 方法不同，（只适用于动画），clearQueue() 能够清除任何排队的函数（通过 .queue() 方法添加到通用 jQuery 队列的任何函数）。

语法

```
$(_selector_).clearQueue(_queueName_)
```

参数	描述
queueName	可选。规定要停止的队列的名称。默认是 "fx"，标准效果队列。

jQuery 效果 - fadeIn() 方法

实例

使用淡入效果来显示一个隐藏的 <p> 元素：

```
$(".btn2").click(function(){
  $("p").fadeIn();
});
```

定义和用法

fadeIn() 方法使用淡入效果来显示被选元素，假如该元素是隐藏的。

语法

```
$(_selector_).fadeIn(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从隐藏到可见的速度。默认为 "normal"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从隐藏到可见的过程中，会逐渐地改变其透明度（这样会创造淡入效果）。
callback	可选。fadeIn 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经显示，则该效果不产生任何变化，除非规定了 callback 函数。

注释：该效果适用于通过 jQuery 隐藏的元素，或在 CSS 中声明 display:none 的元素（但不适用于 visibility:hidden 的元素）。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来对淡入或淡出元素。

jQuery 效果 - fadeOut() 方法

实例

使用淡出效果来隐藏一个 <p> 元素：

```
$(".btn1").click(function(){
  $("p").fadeOut();
});
```

定义和用法

fadeOut() 方法使用淡出效果来隐藏被选元素，假如该元素是隐藏的。

语法

```
$(_selector_).fadeOut(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从可见到隐藏的速度。默认为 "normal"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从可见到隐藏的过程中，会逐渐地改变其透明度（这样会创造淡出效果）。
callback	可选。fadeOut 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经隐藏，则该效果不产生任何变化，除非规定了 callback 函数。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来对淡入或淡出元素。

jQuery 效果 - fadeTo() 方法

实例

使用淡出效果来隐藏一个 <p> 元素：

```
$(".btn1").click(function(){
  $("p").fadeTo(1000,0.4);
});
```

定义和用法

fadeTo() 方法将被选元素的不透明度逐渐地改变为指定的值。

语法

```
$(_selector_).fadeTo(_speed_,_opacity_,_callback_)
```

参数	描述
speed	可选。规定元素从当前透明度到指定透明度的速度。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"
opacity	必需。规定要淡入或淡出的透明度。必须是介于 0.00 与 1.00 之间的数字。
callback	可选。fadeTo 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

jQuery 效果 - hide() 方法

实例

隐藏可见的 <p> 元素：

```
$(".btn1").click(function(){
  $("p").hide();
});
```

定义和用法

如果被选的元素已被显示，则隐藏该元素。

语法

```
$(_selector_).hide(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从可见到隐藏的速度。默认为 "0"。可能的值：毫秒（比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从可见到隐藏的过程中，会逐渐地改变其高度、宽度、外边距、内边距和透明度。
callback	可选。hide 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经是完全可见，则该效果不产生任何变化，除非规定了 callback 函数。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来隐藏和显示元素。

使用 speed 和 callback 参数

使用 speed 和 callback 参数来隐藏和显示元素。

jQuery 效果 - show() 方法

实例

显示出隐藏的 <p> 元素。

```
$(".btn2").click(function(){
  $("p").show();
});
```

定义和用法

如果被选元素已被隐藏，则显示这些元素：

语法

```
$(_selector_).show(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从隐藏到完全可见的速度。默认为 "0"。可能的值：毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从隐藏到完全可见的过程中，会逐渐地改变其高度、宽度、外边距、内边距和透明度。
callback	可选。show 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经是完全可见，则该效果不产生任何变化，除非规定了 callback 函数。

注释：该效果适用于通过 jQuery 隐藏的元素，或在 CSS 中声明 display:none 的元素（但不适用于 visibility:hidden 的元素）。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来隐藏和显示元素。

使用 speed 和 callback 参数

使用 speed 和 callback 参数来隐藏和显示元素。

jQuery 效果 - slideDown() 方法

实例

以滑动方式显示隐藏的 <p> 元素：

```
$(".btn2").click(function(){
  $("p").slideDown();
});
```

定义和用法

slideDown() 方法通过使用滑动效果，显示隐藏的被选元素。

语法

```
$(_selector_).slideDown(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从隐藏到可见的速度（或者相反）。默认为 "normal"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从隐藏到可见的过程中，会逐渐地改变其高度。
callback	可选。slideDown 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经是完全可见，则该效果不产生任何变化，除非规定了 callback 函数。

注释：该效果适用于通过 jQuery 隐藏的元素，或在 CSS 中声明 display:none 的元素（但不适用于 visibility:hidden 的元素）。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来隐藏和显示元素。

jQuery 效果 - slideToggle() 方法

实例

通过使用滑动效果，在显示和隐藏状态之间切换 <p> 元素：

```
$(".btn1").click(function(){
  $("p").slideToggle();
});
```

定义和用法

slideToggle() 方法通过使用滑动效果（高度变化）来切换元素的可见状态。

如果被选元素是可见的，则隐藏这些元素，如果被选元素是隐藏的，则显示这些元素。

语法

```
$(_selector_).slideToggle(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从隐藏到可见的速度（或者相反）。默认为 "normal"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素在切换的过程中，会逐渐地改变其高度（这样会创造滑动效果）。
callback	可选。toggle 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经隐藏，则该效果不产生任何变化，除非规定了 callback 函数。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来切换上下滑动地显示和隐藏元素。

jQuery 效果 - slideUp() 方法

实例

以滑动方式隐藏 <p> 元素：

```
$(".btn1").click(function(){
  $("p").slideUp();
});
```

定义和用法

通过使用滑动效果，隐藏被选元素，如果元素已显示出来的话。

语法

```
$(_selector_).slideUp(_speed_,_callback_)
```

参数	描述
speed	可选。规定元素从可见到隐藏的速度（或者相反）。默认为 "normal"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从可见到隐藏的过程中，会逐渐地改变其高度（这样会创造滑动效果）。
callback	可选。slideUp 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。

提示和注释

提示：如果元素已经隐藏，则该效果不产生任何变化，除非规定了 callback 函数。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来隐藏和显示元素。

jQuery 效果 - stop() 方法

实例

停止当前正在运行的动画：

```
$("#stop").click(function(){
  $("#box").stop();
});
```

定义和用法

stop() 方法停止当前正在运行的动画。

语法

```
$(_selector_).stop(_stopAll_,_goToEnd_)
```

参数	描述
stopAll	可选。规定是否停止被选元素的所有加入队列的动画。
goToEnd	可选。规定是否允许完成当前的动画。该参数只能在设置了 stopAll 参数时使用。

亲自试一试 - 实例

停止动画队列

停止被选元素的所有加入队列的动画。

在当前动画完成后停止动画队列

停止被选元素的所有加入队列的动画，但允许完成当前动画。

jQuery 效果 - toggle() 方法

实例

切换 <p> 元素的显示与隐藏状态：

```
$(".btn1").click(function(){
  $("p").hide();
});
```

定义和用法

toggle() 方法切换元素的可见状态。

如果被选元素可见，则隐藏这些元素，如果被选元素隐藏，则显示这些元素。

语法

```
$(_selector_).toggle(_speed_,_callback_,_switch_)
```

参数	描述
speed	可选。规定元素从可见到隐藏的速度（或者相反）。默认为 "0"。可能的值： 毫秒 （比如 1500） "slow" "normal" "fast"在设置速度的情况下，元素从可见到隐藏的过程中，会逐渐地改变其高度、 宽度、外边距、内边距和透明度。如果设置此参数，则无法使用 switch 参数。
callback	可选。toggle 函数执行完之后，要执行的函数。如需学习更多有关 callback 的内容，请访问我们的 jQuery Callback 这一章。除非设置了 speed 参数，否则不能设置该参数。
switch	可选。布尔值。规定 toggle 是否隐藏或显示所有被选元素。 True - 显示所有元素 False - 隐藏所有元素如果设置此参数，则无法使用 speed 和 callback 参数。

提示和注释

注释：该效果适用于通过 jQuery 隐藏的元素，或在 CSS 中声明 display:none 的元素（但不适用于 visibility:hidden 的元素）。

亲自试一试 - 实例

使用 speed 参数

使用 speed 参数来隐藏和显示元素。

使用 switch 参数

使用 switch 参数来显示所有隐藏的段落。

jQuery 参考手册 - 文档操作

jQuery 文档操作方法

这些方法对于 XML 文档和 HTML 文档均是适用的，除了：html()。

方法	描述
addClass()	向匹配的元素添加指定的类名。
after()	在匹配的元素之后插入内容。
append()	向匹配元素集合中的每个元素结尾插入由参数指定的内容。
appendTo()	向目标结尾插入匹配元素集合中的每个元素。
attr()	设置或返回匹配元素的属性和值。
before()	在每个匹配的元素之前插入内容。
clone()	创建匹配元素集合的副本。
detach()	从 DOM 中移除匹配元素集合。
empty()	删除匹配的元素集合中所有的子节点。
hasClass()	检查匹配的元素是否拥有指定的类。
html()	设置或返回匹配的元素集合中的 HTML 内容。
insertAfter()	把匹配的元素插入到另一个指定的元素集合的后面。
insertBefore()	把匹配的元素插入到另一个指定的元素集合的前面。
prepend()	向匹配元素集合中的每个元素开头插入由参数指定的内容。
prependTo()	向目标开头插入匹配元素集合中的每个元素。
remove()	移除所有匹配的元素。
removeAttr()	从所有匹配的元素中移除指定的属性。
removeClass()	从所有匹配的元素中删除全部或者指定的类。
replaceAll()	用匹配的元素替换所有匹配到的元素。
replaceWith()	用新内容替换匹配的元素。
text()	设置或返回匹配元素的内容。
toggleClass()	从匹配的元素中添加或删除一个类。
unwrap()	移除并替换指定元素的父元素。
val()	设置或返回匹配元素的值。
wrap()	把匹配的元素用指定的内容或元素包裹起来。
wrapAll()	把所有匹配的元素用指定的内容或元素包裹起来。
wrapinner()	将每一个匹配的元素的子内容用指定的内容或元素包裹起来。

jQuery 属性操作 - addClass() 方法

实例

向第一个 p 元素添加一个类：

```
$("button").click(function(){
  $("p:first").addClass("intro");
});
```

定义和用法

addClass() 方法向被选元素添加一个或多个类。

该方法不会移除已存在的 class 属性，仅仅添加一个或多个 class 属性。

提示：如需添加多个类，请使用空格分隔类名。

语法

```
$(selector).addClass(_class_)
```

参数	描述
class	必需。规定一个或多个 class 名称。

使用函数来添加类

使用函数向被选元素添加类。

语法

```
$(selector).addClass(_function_(_index_,_oldclass)_)
```

参数	描述
function(index,oldclass)	必需。规定返回一个或多个待添加类名的函数。 *index* - 可选。选择器的 index 位置。 *class* - 可选。选择器的旧的类名。

亲自试一试 - 实例

向元素添加两个类

如何向被选元素添加两个 class。

改变元素的类

如任何使用 addClass() 和 removeClass() 来移除 class，并添加新的 class。

jQuery 文档操作 - after() 方法

实例

在每个 p 元素后插入内容：

```
$("button").click(function(){
  $("p").after("<p>Hello world!</p>");
});
```

定义和用法

after() 方法在被选元素后插入指定的内容。

语法

```
$(selector).after(_content_)
```

参数	描述
content	必需。规定要插入的内容（可包含 HTML 标签）。

使用函数来插入内容

使用函数在被选元素之后插入指定的内容。

语法

```
$(selector).after(_function_(_index_))
```

参数	描述
function(*index*)	必需。规定返回待插入内容的函数。 *index* - 可选。接收选择器的 index 位置。

jQuery 文档操作 - append() 方法

实例

在每个 p 元素结尾插入内容：

```
$("button").click(function(){
  $("p").append(" <b>Hello world!</b>");
});
```

定义和用法

append() 方法在被选元素的结尾（仍然在内部）插入指定内容。

提示：append() 和 appendTo() 方法执行的任务相同。不同之处在于：内容的位置和选择器。

语法

```
$(selector).append(_content_)
```

参数	描述
content	必需。规定要插入的内容（可包含 HTML 标签）。

使用函数来附加内容

使用函数在指定元素的结尾插入内容。

语法

```
$(selector).append(_function(index,html)_)
```

参数	描述
function(index,html)	必需。规定返回待插入内容的函数。 *index* - 可选。接收选择器的 index 位置。 *html* - 可选。接收选择器的当前 HTML。

jQuery 文档操作 - appendTo() 方法

实例

在每个 p 元素结尾插入内容：

```
$("button").click(function(){
  $("<b>Hello World!</b>").appendTo("p");
});
```

定义和用法

appendTo() 方法在被选元素的结尾（仍然在内部）插入指定内容。

提示：append() 和 appendTo() 方法执行的任务相同。不同之处在于：内容和选择器的位置，以及 append() 能够使用函数来附加内容。

语法

```
$(_content_).appendTo(_selector_)
```

参数	描述
content	必需。规定要插入的内容（可包含 HTML 标签）。
selector	必需。规定把内容追加到哪个元素上。

jQuery 属性操作 - attr() 方法

实例

改变图像的 width 属性：

```
$("button").click(function(){
  $("img").attr("width","180");
});
```

定义和用法

attr() 方法设置或返回被选元素的属性值。

根据该方法不同的参数，其工作方式也有所差异。

返回属性值

返回被选元素的属性值。

语法

```
$(_selector_).attr(_attribute_)
```

参数	描述
attribute	规定要获取其值的属性。

设置属性/值

设置被选元素的属性和值。

语法

```
$(_selector_).attr(_attribute_,_value_)
```

参数	描述
attribute	规定属性的名称。
value	规定属性的值。

使用函数来设置属性/值

设置被选元素的属性和值。

语法

```
$(_selector_).attr(_attribute_,_function(index,oldvalue)_)
```

参数	描述
attribute	规定属性的名称。
function(index,oldvalue)	规定返回属性值的函数。该函数可接收并使用选择器的 index 值和当前属性值。

设置多个属性/值对

为被选元素设置一个以上的属性和值。

语法

```
$(_selector_).attr({_attribute:value_, _attribute:value_ ...})
```

参数	描述
attribute:value	规定一个或多个属性/值对。

jQuery 文档操作 - before() 方法

实例

在每个 p 元素前插入内容：

```
$("button").click(function(){
  $("p").before("<p>Hello world!</p>");
});
```

定义和用法

before() 方法在被选元素前插入指定的内容。

语法

```
$(selector).before(_content_)
```

参数	描述
content	必需。规定要插入的内容（可包含 HTML 标签）。

使用函数来插入内容

使用函数在指定的元素前面插入内容。

语法

```
$(selector).before(_function(index)_)
```

参数	描述
function(index)	必需。规定返回待插入内容的函数。 *index* - 可选。接收选择器的 index 位置。

jQuery 文档操作 - clone() 方法

实例

克隆并追加一个 p 元素：

```
$("button").click(function(){
  $("body").append($("p").clone());
});
```

定义和用法

clone() 方法生成被选元素的副本，包含子节点、文本和属性。

语法

```
$(_selector_).clone(_includeEvents_)
```

参数	描述
includeEvents	可选。布尔值。规定是否复制元素的所有事件处理。默认地，副本中不包含事件处理器。

亲自试一试 - 实例

复制一个元素，包括事件处理器

使用 clone() 方法来复制元素，包括其事件处理器。

jQuery 文档操作 - detach() 方法

实例

移除所有 p 元素：

```
$("button").click(function(){
  $("p").detach();
});
```

定义和用法

detach() 方法移除被选元素，包括所有文本和子节点。

这个方法会保留 jQuery 对象中的匹配的元素，因而可以在将来再使用这些匹配的元素。

detach() 会保留所有绑定的事件、附加的数据，这一点与 remove() 不同。

语法

```
$(_selector_).detach()
```

亲自试一试 - 实例

移动元素

使用 detach() 方法来移动元素。

删除并恢复一个元素

使用 detach() 方法来删除并恢复一个元素。

移动元素并保留其 click 事件

使用 detach() 方法来移动元素，并保留元素的 jQuery 数据。

jQuery 文档操作 - empty() 方法

实例

移除 p 元素的内容：

```
$(".btn1").click(function(){
  $("p").empty();
});
```

定义和用法

empty() 方法从被选元素移除所有内容，包括所有文本和子节点。

语法

```
$(_selector_).empty()
```

亲自试一试 - 实例

移除元素的内容

使用 empty() 方法从元素移除内容。

jQuery 属性操作 - hasClass() 方法

实例

检查第一个 <p> 元素是否包含 "intro" 类：

```
$("button").click(function(){
  alert($("p:first").hasClass("intro"));
});
```

定义和用法

hasClass() 方法检查被选元素是否包含指定的 class。

语法

```
$(_selector_).hasClass(_class_)
```

参数	描述
class	必需。规定需要在指定元素中查找的类。

jQuery 文档操作 - html() 方法

实例

设置所有 p 元素的内容：

```
$(".btn1").click(function(){
  $("p").html("Hello <b>world</b>!");
});
```

定义和用法

html() 方法返回或设置被选元素的内容 (inner HTML)。

如果该方法未设置参数，则返回被选元素的当前内容。

返回元素内容

当使用该方法返回一个值时，它会返回第一个匹配元素的内容。

语法

```
$(selector).html()
```

设置元素内容

当使用该方法设置一个值时，它会覆盖所有匹配元素的内容。

语法

```
$(_selector_).html(_content_)
```

参数	描述
content	可选。规定被选元素的新内容。该参数可包含 HTML 标签。

使用函数来设置元素内容

使用函数来设置所有匹配元素的内容。

语法

```
$(_selector_).html(_function(index,oldcontent)_)
```

参数	描述
function(index,oldcontent)	规定一个返回被选元素的新内容的函数。 *index* - 可选。接收选择器的 index 位置。 *oldcontent* - 可选。接收选择器的当前内容。

jQuery 文档操作 - insertAfter() 方法

实例

在每个 p 元素之后插入 span 元素：

```
$("button").click(function(){
  $("<span>Hello world!</span>").insertAfter("p");
});
```

定义和用法

insertAfter() 方法在被选元素之后插入 HTML 标记或已有的元素。

注释：如果该方法用于已有元素，这些元素会被从当前位置移走，然后被添加到被选元素之后。

语法

```
$(_content_).insertAfter(_selector_)
```

参数	描述
content	必需。规定要插入的内容。可能的值：选择器表达式 HTML 标记
selector	必需。规定在何处插入被选元素。

亲自试一试 - 实例

插入已有的元素

如何使用 insertAfter() 方法在每个被选元素之后插入已存在的元素。

jQuery 文档操作 - insertBefore() 方法

实例

在每个 p 元素之前插入 span 元素：

```
$("button").click(function(){
  $("<span>Hello world!</span>").insertBefore("p");
});
```

定义和用法

insertBefore() 方法在被选元素之前插入 HTML 标记或已有的元素。

注释：如果该方法用于已有元素，这些元素会被从当前位置移走，然后被添加到被选元素之前。

语法

```
$(_content_).insertBefore(_selector_)
```

参数	描述
content	必需。规定要插入的内容。可能的值： 选择器表达式 HTML 标记
selector	必需。规定在何处插入被选元素。

亲自试一试 - 实例

插入已有的元素

如何使用 insertAfter() 方法在每个被选元素之前插入已存在的元素。

jQuery 文档操作 - prepend() 方法

实例

在 p 元素的开头插入内容：

```
$(".btn1").click(function(){
  $("p").prepend("<b>Hello world!</b>");
});
```

定义和用法

prepend() 方法在被选元素的开头（仍位于内部）插入指定内容。

提示：prepend() 和 prependTo() 方法作用相同。差异在于语法：内容和选择器的位置，以及 prependTo() 无法使用函数来插入内容。

语法

```
$(_selector_).prepend(_content_)
```

参数	描述
content	必需。规定要插入的内容（可包含 HTML 标签）。

使用函数来附加内容

使用函数在被选元素的开头插入指定的内容。

语法

```
$(_selector_).prepend(_function(index,html)_)
```

参数	描述
function(index,html)	必需。规定返回待插入内容的函数。 `index` - 可选。接受选择器的 index 位置。 `html` - 可选。接受选择器的当前 HTML。

jQuery 文档操作 - prependTo() 方法

实例

在每个 p 元素的开头插入内容：

```
$(".btn1").click(function(){
  $("<b>Hello World!</b>").prependTo("p");
});
```

定义和用法

prependTo() 方法在被选元素的开头（仍位于内部）插入指定内容。

提示：prepend() 和 prependTo() 方法作用相同。差异在于语法：内容和选择器的位置，以及 prepend() 能够使用函数来插入内容。

语法

```
$(_content_).prependTo(_selector_)
```

参数	描述
content	必需。规定要插入的内容（可包含 HTML 标签）。
selector	必需。规定在何处插入内容。

jQuery 文档操作 - remove() 方法

实例

移除所有 <p> 元素：

```
$("button").click(function(){
  $("p").remove();
});
```

定义和用法

remove() 方法移除被选元素，包括所有文本和子节点。

该方法不会把匹配的元素从 jQuery 对象中删除，因而可以在将来再使用这些匹配的元素。

但除了这个元素本身得以保留之外，remove() 不会保留元素的 jQuery 数据。其他的比如绑定的事件、附加的数据等都会被移除。这一点与 detach() 不同。

语法

```
$(_selector_).remove()
```

亲自试一试 - 实例

移动元素

使用 remove() 方法来移动元素。

jQuery 属性操作 - removeAttr() 方法

实例

从任何 p 元素中移除 id 属性：

```
$("button").click(function(){
  $("p").removeAttr("id");
});
```

定义和用法

removeAttr() 方法从被选元素中移除属性。

语法

```
$(_selector_).removeAttr(_attribute_)
```

参数	描述
attribute	必需。规定从指定元素中移除的属性。

jQuery 属性操作 - removeClass() 方法

实例

移除所有 <p> 的 "intro" 类：

```
$("button").click(function(){
  $("p:first").removeClass("intro");
});
```

定义和用法

removeClass() 方法从被选元素移除一个或多个类。

注释：如果没有规定参数，则该方法将从被选元素中删除所有类。

语法

```
$(_selector_).removeClass(_class_)
```

参数	描述
class	可选。规定要移除的 class 的名称。如需移除若干类，请使用空格来分隔类名。如果不设置该参数，则会移除所有类。

使用函数来移除类

使用函数来删除被选元素中的类。

```
$(_selector_).removeClass(_function(index,oldclass)_)
```

参数	描述
function(index,oldclass)	必需。通过运行函数来移除指定的类。 `index` - 可选。接受选择器的 index 位置。 `html` - 可选。接受选择器的旧的类值。

亲自试一试 - 实例

改变元素的类

如何使用 addClass() 和 removeClass() 来移除一个类，并添加一个新的类。

jQuery 文档操作 - replaceAll() 方法

实例

用粗体文本替换每个段落：

```
$(".btn1").click(function(){
   $("p").replaceAll("<b>Hello world!</b>");
});
```

定义和用法

replaceAll() 方法用指定的 HTML 内容或元素替换被选元素。

提示：replaceAll() 与 replaceWith() 作用相同。差异在于语法：内容和选择器的位置，以及 replaceWith() 能够使用函数进行替换。

语法

```
$(_content_).replaceAll(_selector_)
```

参数	描述
content	必需。规定替换被选元素的内容。可能的值：`HTML 代码` - 比如 (" ") `新元素` - 比如 (document.createElement("div")) `已存在的元素` - 比如 ($(".div1")) 已存在的元素不会被移动，只会被复制，并包裹被选元素。
selector	必需。规定要替换的元素。

亲自试一试 - 实例

使用新元素来替换元素

使用 document.createElement() 来创建一个新的 DOM 元素，然后用它替换被选元素。

jQuery 文档操作 - replaceWith() 方法

实例

用粗体文本替换每个段落：

```
$(".btn1").click(function(){
   $("p").replaceWith("<b>Hello world!</b>");
});
```

定义和用法

replaceWith() 方法用指定的 HTML 内容或元素替换被选元素。

提示：replaceWith() 与 replaceAll() 作用相同。差异在于语法：内容和选择器的位置，以及 replaceAll() 无法使用函数进行替换。

语法

```
$(_selector_).replaceWith(_content_)
```

参数	描述
content	必需。规定替换被选元素的内容。可能的值：`HTML 代码` - 比如 (" ") `新元素` - 比如 (document.createElement("div")) `已存在的元素` - 比如 ($(".div1")) 已存在的元素不会被移动，只会被复制，并包裹被选元素。
selector	必需。规定要替换的元素。

使用函数来替换元素

使用函数把被选元素替换为新内容。

语法

```
$(_selector_).replaceWith(_function()_)
```

参数	描述
function()	必需。返回待替换被选元素的新内容的函数。

亲自试一试 - 实例

使用新元素来替换元素

使用 document.createElement() 来创建一个新的 DOM 元素，然后用它替换被选元素。

jQuery 文档操作 - text() 方法

实例

设置所有 <p> 元素的内容：

```
$(".btn1").click(function(){
  $("p").text("Hello <b>world</b>!");
});
```

定义和用法

text() 方法方法设置或返回被选元素的文本内容。

返回文本内容

当该方法用于返回一个值时，它会返回所有匹配元素的组合的文本内容（会删除 HTML 标记）。

语法

```
$(_selector_).text()
```

设置文本内容

当该方法用于设置值时，它会覆盖被选元素的所有内容。

```
$(_selector_).text(_content_)
```

参数	描述
content	规定被选元素的新文本内容。注释：特殊字符会被编码。

使用函数设置文本内容

使用函数设置所有被选元素的文本内容。

语法

```
$(_selector_).text(_function(index,oldcontent)_)
```

参数	描述
function(index,oldcontent)	必需。规定返回被选元素的新文本内容的函数。 `index` - 可选。接受选择器的 index 位置。 `html` - 可选。接受选择器的当前内容。

jQuery 属性操作 - toggleClass() 方法

实例

对设置和移除所有 <p> 元素的 "main" 类进行切换：

```
$("button").click(function(){
  $("p").toggleClass("main");
});
```

定义和用法

toggleClass() 对设置或移除被选元素的一个或多个类进行切换。

该方法检查每个元素中指定的类。如果不存在则添加类，如果已设置则删除之。这就是所谓的切换效果。

不过，通过使用 "switch" 参数，您能够规定只删除或只添加类。

语法

```
$(_selector_).toggleClass(_class_,_switch_)
```

参数	描述
class	必需。规定添加或移除 class 的指定元素。如需规定若干 class，请使用空格来分隔类名。
switch	可选。布尔值。规定是否添加或移除 class。

使用函数来切换类

```
$(_selector_).toggleClass(_function(index,class)_,_switch_)
```

参数	描述
function(index,class)	必需。规定返回需要添加或删除的一个或多个类名的函数。 `index` - 可选。接受选择器的 index 位置。 `class` - 可选。接受选择器的当前的类。
switch	可选。布尔值。规定是否添加(true)或移除(false)类。

jQuery 文档操作 - unwrap() 方法

实例

删除所有 <p> 元素的父元素：

```
$("button").click(function(){
  $("p").unwrap();
});
```

定义和用法

unwrap() 方法删除被选元素的父元素。

语法

```
$(_selector_).unwrap()
```

亲自试一试 - 实例

包裹或解开元素

对包裹和解开元素的任务进行切换。

jQuery 属性操作 - val() 方法

实例

设置输入域的值：

```
$("button").click(function(){
  $(":text").val("Hello World");
});
```

定义和用法

val() 方法返回或设置被选元素的值。

元素的值是通过 value 属性设置的。该方法大多用于 input 元素。

如果该方法未设置参数，则返回被选元素的当前值。

语法

```
$(_selector_).val(_value_)
```

参数	描述
value	可选。规定被选元素的新内容。

返回 Value 属性

返回第一个匹配元素的 value 属性的值。

语法

```
$(_selector_).val()
```

设置 Value 属性的值

```
$(_selector_).val(_value_)
```

参数	描述
value	设置 Value 属性的值。

使用函数设置 Value 属性的值

```
$(_selector_).val(_function(index,oldvalue)_)
```

参数	描述
function(index,oldvalue)	规定返回要设置的值的函数。 *index* - 可选。接受选择器的 index 位置。 *oldvalue* - 可选。接受选择器的当前 Value 属性。

jQuery 文档操作 - wrap() 方法

实例

在 <div> 元素中包裹每个段落：

```
$(".btn1").click(function(){
   $("p").wrap("<div></div>");
});
```

定义和用法

wrap() 方法把每个被选元素放置在指定的 HTML 内容或元素中。

语法

```
$(_selector_).wrap(_wrapper_)
```

参数	描述
wrapper	必需。规定包裹被选元素的内容。可能的值: `HTML 代码` - 比如 (" ") `新元素` - 比如 (document.createElement("div")) `已存在的元素` - 比如 ($(".div1")) 已存在的元素不会被移动，只会被复制，并包裹被选元素。

使用函数来包裹元素

使用函数来规定在每个被选元素周围包裹的内容。

语法

```
$(_selector_).wrap(_function()_)
```

参数	描述
function()	必需。规定返回包裹元素的函数。

亲自试一试 - 实例

使用新元素来包裹

创建一个新的 DOM 元素来包裹每个被选元素。

包裹或解开元素

对包裹和解开元素的任务进行切换。

jQuery 文档操作 - wrapAll() 方法

实例

在 <div> 中包裹所有段落：

```
$(".btn1").click(function(){
   $("p").wrapAll("<div></div>");
});
```

定义和用法

wrapAll() 在指定的 HTML 内容或元素中放置所有被选的元素。

语法

```
$(_selector_).wrapAll(_wrapper_)
```

参数	描述
wrapper	必需。规定包裹被选元素的内容。可能的值: `HTML 代码` - 比如 (" ") `新的 DOM 元素` - 比如 (document.createElement("div")) `已存在的元素` - 比如 ($(".div1")) 已存在的元素不会被移动，只会被复制，并包裹被选元素。

亲自试一试 - 实例

使用新元素来包裹

创建一个新的 DOM 元素来包裹每个被选元素。

jQuery 文档操作 - wrapInner() 方法

实例

在每个 p 元素的内容上包围 b 元素：

```
$(".btn1").click(function(){
   $("p").wrapInner("<b></b>");
});
```

定义和用法

wrapInner() 方法使用指定的 HTML 内容或元素，来包裹每个被选元素中的所有内容 (inner HTML)。

语法

```
$(_selector_).wrapInner(_wrapper_)
```

参数	描述
wrapper	必需。规定包围在被选元素的内容周围的内容。可能的值: `HTML 代码` - 比如 (" ") `新的 DOM 元素` - 比如 (document.createElement("div")) `已存在的元素` - 比如 ($(".div1")) 已存在的元素不会被移动，只会被复制，并包裹被选元素。

使用函数包裹内容

使用函数来规定包围在每个被选元素周围的内容。

语法

```
$(_selector_).wrapInner(_function()_)
```

参数	描述
function()	必需。规定返回包围元素的函数。

亲自试一试 - 实例

使用新元素来包裹

创建一个新的 DOM 元素来包裹每个被选元素。

jQuery 参考手册 - 属性操作

jQuery 属性操作方法

下面列出的这些方法获得或设置元素的 DOM 属性。

这些方法对于 XML 文档和 HTML 文档均是适用的，除了：html()。

方法	描述
addClass()	向匹配的元素添加指定的类名。
attr()	设置或返回匹配元素的属性和值。
hasClass()	检查匹配的元素是否拥有指定的类。
html()	设置或返回匹配的元素集合中的 HTML 内容。
removeAttr()	从所有匹配的元素中移除指定的属性。
removeClass()	从所有匹配的元素中删除全部或者指定的类。
toggleClass()	从匹配的元素中添加或删除一个类。
val()	设置或返回匹配元素的值。

注释：中也列出了以上方法。本参考页的作用是方便用户单独查阅有关属性操作方面的方法。

jQuery 参考手册 - CSS 操作

jQuery CSS 操作函数

下面列出的这些方法设置或返回元素的 CSS 相关属性。

CSS 属性	描述
css()	设置或返回匹配元素的样式属性。
height()	设置或返回匹配元素的高度。
offset()	返回第一个匹配元素相对于文档的位置。
offsetParent()	返回最近的定位祖先元素。
position()	返回第一个匹配元素相对于父元素的位置。
scrollLeft()	设置或返回匹配元素相对滚动条左侧的偏移。
scrollTop()	设置或返回匹配元素相对滚动条顶部的偏移。
width()	设置或返回匹配元素的宽度。

参阅

教程：CSS 教程

参考手册：CSS 参考手册

jQuery CSS 操作 - css() 方法

实例

设置 <p> 元素的颜色：

```
$(".btn1").click(function(){
  $("p").css("color","red");
});
```

定义和用法

css() 方法返回或设置匹配的元素的一个或多个样式属性。

返回 CSS 属性值

返回第一个匹配元素的 CSS 属性值。

注释：当用于返回一个值时，不支持简写的 CSS 属性（比如 "background" 和 "border"）。

```
$(_selector_).css(_name_)
```

参数	描述
name	必需。规定 CSS 属性的名称。该参数可包含任何 CSS 属性。比如 "color"。

实例

取得第一个段落的 color 样式属性的值：

```
$("p").css("color");
```

设置 CSS 属性

设置所有匹配元素的指定 CSS 属性。

```
$(_selector_).css(_name_,_value_)
```

参数	描述
name	必需。规定 CSS 属性的名称。该参数可包含任何 CSS 属性，比如 "color"。
value	可选。规定 CSS 属性的值。该参数可包含任何 CSS 属性值，比如 "red"。如果设置了空字符串值，则从元素中删除指定属性。

实例

将所有段落的颜色设为红色：

```
$("p").css("color","red");
```

使用函数来设置 CSS 属性

设置所有匹配的元素中样式属性的值。

此函数返回要设置的属性值。接受两个参数，index 为元素在对象集合中的索引位置，value 是原先的属性值。

```
$(_selector_).css(_name_,_function(index,value)_)
```

参数	描述
name	必需。规定 CSS 属性的名称。该参数可包含任何 CSS 属性，比如 "color"。
function(index,value)	规定返回 CSS 属性新值的函数。 `index` - 可选。接受选择器的 index 位置 `oldvalue` - 可选。接受 CSS 属性的当前值。

实例 1

将所有段落的颜色设为红色：

```
$("button").click(function(){
    $("p").css("color",function(){return "red";});
    });
```

实例 2

逐渐增加 div 的宽度：

```
$("div").click(function() {
  $(this).css(
    "width", function(index, value) {return parseFloat(value) * 1.2
  );
});
```

设置多个 CSS 属性/值对

```
$(_selector_).css({_property:value_, _property:value_, ...})
```

把“名/值对”对象设置为所有匹配元素的样式属性。

这是一种在所有匹配的元素上设置大量样式属性的最佳方式。

参数	描述
{*property:value*}	必需。规定要设置为样式属性的“名称/值对”对象。该参数可包含若干对 CSS 属性名称/值。比如 {"color":"red","font-weight":"bold"}

实例

```
$("p").css({
  "color":"white",
  "background-color":"#98bf21",
  "font-family":"Arial",
  "font-size":"20px",
  "padding":"5px"
  });
```

jQuery CSS 操作 - height() 方法

实例

设置 <p> 元素的高度：

```
$(".btn1").click(function(){
  $("p").height(50);
});
```

定义和用法

height() 方法返回或设置匹配元素的高度。

返回高度

返回第一个匹配元素的高度。

如果不为该方法设置参数，则返回以像素计的匹配元素的高度。

语法

```
$(_selector_).height()
```

设置高度

设置所有匹配元素的高度。

语法

```
$(_selector_).height(_length_)
```

参数	描述
length	可选。规定元素的高度。如果没有规定长度单位，则使用默认的 px 单位。

使用函数来设置高度

使用函数来设置所有匹配元素的高度。

语法

```
$(_selector_).height(_function(index,oldheight)_)
```

参数	描述
function(index,oldheight)	规定返回被选元素新高度的函数。 `index` - 可选。接受选择器的 index 位置 `oldvalue` - 可选。接受选择器的当前值。

亲自试一试 - 实例

[获得文档和窗口元素的高度]

使用 height() 方法来获得 document 和 window 元素的当前高度。

[使用 em 和 % 值来设置高度]

使用指定的长度单位来设置元素的高度。

jQuery CSS 操作 - offset() 方法

实例

获得 <p> 元素当前的偏移：

```
$(".btn1").click(function(){
  x=$("p").offset();
  $("#span1").text(x.left);
  $("#span2").text(x.top);
});
```

定义和用法

offset() 方法返回或设置匹配元素相对于文档的偏移（位置）。

返回偏移坐标

返回第一个匹配元素的偏移坐标。

该方法返回的对象包含两个整型属性：top 和 left，以像素计。此方法只对可见元素有效。

语法

```
$(_selector_).offset()
```

设置偏移坐标

设置所有匹配元素的偏移坐标。

语法

```
$(_selector_).offset(_value_)
```

参数	描述
value	必需。规定以像素计的 top 和 left 坐标。可能的值： 值对， 比如 {top:100,left:0}； 带有 top 和 left 属性的对象。

使用函数来设置偏移坐标

使用函数来设置所有匹配元素的偏移坐标。

语法

```
$(_selector_).offset(_function(index,oldoffset)_)
```

参数	描述
function(index,oldoffset)	规定返回被选元素新偏移坐标的函数。 `index` - 可选。接受选择器的 index 位置 `oldvalue` - 可选。接受选择器的当前坐标。

亲自试一试 - 实例

使用对象来为对象设置新的 offset 值

使用新对象中的坐标来定位元素。

使用另一个元素的位置来为元素设置新的 offset 值

使用已有对象的位置来定位元素。

jQuery CSS 操作 - offsetParent() 方法

实例

设置最近的祖先定位元素的背景颜色：

```
$("button").click(function(){
  $("p").offsetParent().css("background-color","red");
});
```

定义和用法

offsetParent() 方法返回最近的祖先定位元素。

定位元素指的是元素的 CSS position 属性被设置为 relative、absolute 或 fixed 的元素。

可以通过 jQuery 设置 position，或者通过 CSS 的 position 属性。

语法

```
$(_selector_).offsetParent()
```

jQuery CSS 操作 - position() 方法

实例

获得 <p> 元素的当前位置：

```
$(".btn1").click(function(){
  x=$("p").position();
  $("#span1").text(x.left);
  $("#span2").text(x.top);
});
```

定义和用法

position() 方法返回匹配元素相对于父元素的位置（偏移）。

该方法返回的对象包含两个整型属性：top 和 left，以像素计。

此方法只对可见元素有效。

语法

```
$(_selector_).position()
```

jQuery CSS 操作 - scrollLeft() 方法

实例

设置 <div> 元素中滚动条的水平偏移：

```
$(".btn1").click(function(){
  $("div").scrollLeft(100);
});
```

定义和用法

scrollLeft() 方法返回或设置匹配元素的滚动条的水平位置。

滚动条的水平位置指的是从其左侧滚动过的像素数。当滚动条位于最左侧时，位置是 0。

返回水平滚动条位置

返回第一个匹配元素的水平滚动条位置。

语法

```
$(_selector_).scrollLeft()
```

设置水平滚动条位置

设置所有匹配元素的水平滚动条位置。

语法

```
$(_selector_).scrollLeft(_position_)
```

参数	描述
position	可选。规定以像素计的新位置。

jQuery CSS 操作 - scrollTop() 方法

实例

设置 <div> 元素中滚动条的垂直偏移：

```
$(".btn1").click(function(){
  $("div").scrollLeft(100);
});
```

定义和用法

scrollTop() 方法返回或设置匹配元素的滚动条的垂直位置。

scroll top offset 指的是滚动条相对于其顶部的偏移。

如果该方法未设置参数，则返回以像素计的相对滚动条顶部的偏移。

语法

```
$(_selector_).scrollTop(_offset_)
```

参数	描述
offset	可选。规定相对滚动条顶部的偏移，以像素计。

提示和注释

注释：该方法对于可见元素和不可见元素均有效。

注释：当用于获取值时，该方法只返回第一个匹配元素的 scroll top offset。

注释：当用于设置值时，该方法设置所有匹配元素的 scroll top offset。

亲自试一试 - 实例

获得当前的 scroll top offset

使用 scrollTop() 方法获得 scroll top offset。

jQuery CSS 操作 - width() 方法

实例

设置 <p> 元素的宽度：

```
$(".btn1").click(function(){
  $("p").width(200);
});
```

定义和用法

width() 方法返回或设置匹配元素的宽度。

返回宽度

返回第一个匹配元素的宽度。

如果不为该方法设置参数，则返回以像素计的匹配元素的宽度。

语法

```
$(_selector_).width()
```

设置宽度

设置所有匹配元素的宽度。

语法

```
$(_selector_).width(_length_)
```

参数	描述
length	可选。规定元素的宽度。如果没有规定长度单位，则使用默认的 px 单位。

使用函数来设置宽度

使用函数来设置所有匹配元素的宽度。

语法

```
$(_selector_).width(_function(index,oldwidth)_)
```

参数	描述
function(index,oldwidth)	规定返回被选元素新宽度的函数。 `index` - 可选。接受选择器的 index 位置 `oldvalue` - 可选。接受选择器的当前值。

亲自试一试 - 实例

获得文档和窗口元素的宽度

使用 width() 方法来获得 document 和 window 元素的当前宽度。

使用 em 和 % 值来设置宽度

使用指定的长度单位来设置元素的宽度。

jQuery 参考手册 - Ajax

jQuery Ajax 操作函数

jQuery 库拥有完整的 Ajax 兼容套件。其中的函数和方法允许我们在不刷新浏览器的情况下从服务器加载数据。

函数	描述
jQuery.ajax()	执行异步 HTTP (Ajax) 请求。
.ajaxComplete()	当 Ajax 请求完成时注册要调用的处理程序。这是一个 Ajax 事件。
.ajaxError()	当 Ajax 请求完成且出现错误时注册要调用的处理程序。这是一个 Ajax 事件。
.ajaxSend()	在 Ajax 请求发送之前显示一条消息。
jQuery.ajaxSetup()	设置将来的 Ajax 请求的默认值。
.ajaxStart()	当首个 Ajax 请求完成开始时注册要调用的处理程序。这是一个 Ajax 事件。
.ajaxStop()	当所有 Ajax 请求完成时注册要调用的处理程序。这是一个 Ajax 事件。
.ajaxSuccess()	当 Ajax 请求成功完成时显示一条消息。
jQuery.get()	使用 HTTP GET 请求从服务器加载数据。
jQuery.getJSON()	使用 HTTP GET 请求从服务器加载 JSON 编码数据。
jQuery.getScript()	使用 HTTP GET 请求从服务器加载 JavaScript 文件，然后执行该文件。
.load()	从服务器加载数据，然后把返回到 HTML 放入匹配元素。
jQuery.param()	创建数组或对象的序列化表示，适合在 URL 查询字符串或 Ajax 请求中使用。
jQuery.post()	使用 HTTP POST 请求从服务器加载数据。
.serialize()	将表单内容序列化为字符串。
.serializeArray()	序列化表单元素，返回 JSON 数据结构数据。

参阅

教程：Ajax 教程

jQuery ajax - ajax() 方法

实例

通过 AJAX 加载一段文本：

jQuery 代码：

```
$(document).ready(function(){
  $("#b01").click(function(){
  htmlobj=$.ajax({url:"/jquery/test1.txt",async:false});
  $("#myDiv").html(htmlobj.responseText);
  });
});
```

HTML 代码：

```
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button id="b01" type="button">Change Content</button>
```

定义和用法

ajax() 方法通过 HTTP 请求加载远程数据。

该方法是 jQuery 底层 AJAX 实现。简单易用的高层实现见 $.get, $.post 等。$.ajax() 返回其创建的 XMLHttpRequest 对象。大多数情况下你无需直接操作该函数，除非你需要操作不常用的选项，以获得更多的灵活性。

最简单的情况下，$.ajax() 可以不带任何参数直接使用。

注意：所有的选项都可以通过 $.ajaxSetup() 函数来全局设置。

语法

```
jQuery.ajax([_settings_])
```

参数	描述
settings	可选。用于配置 Ajax 请求的键值对集合。可以通过 $.ajaxSetup() 设置任何选项的默认值。

参数

options

类型：Object

可选。AJAX 请求设置。所有选项都是可选的。

async

类型：Boolean

默认值: true。默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。

注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。

beforeSend(XHR)

类型：Function

发送请求前可修改 XMLHttpRequest 对象的函数，如添加自定义 HTTP 头。

XMLHttpRequest 对象是唯一的参数。

这是一个 Ajax 事件。如果返回 false 可以取消本次 ajax 请求。

cache

类型：Boolean

默认值: true，dataType 为 script 和 jsonp 时默认为 false。设置为 false 将不缓存此页面。

jQuery 1.2 新功能。

complete(XHR, TS)

类型：Function

请求完成后回调函数 (请求成功或失败之后均调用)。

参数： XMLHttpRequest 对象和一个描述请求类型的字符串。

这是一个 Ajax 事件。

contentType

类型：String

默认值: "application/x-www-form-urlencoded"。发送信息至服务器时内容编码类型。

默认值适合大多数情况。如果你明确地传递了一个 content-type 给 $.ajax() 那么它必定会发送给服务器（即使没有数据要发送）。

context

类型：Object

这个对象用于设置 Ajax 相关回调函数的上下文。也就是说，让回调函数内 this 指向这个对象（如果不设定这个参数，那么 this 就指向调用本次 AJAX 请求时传递的 options 参数）。比如指定一个 DOM 元素作为 context 参数，这样就设置了 success 回调函数的上下文为这个 DOM 元素。

就像这样：

```
$.ajax({ url: "test.html", context: document.body, success: functio
        $(this).addClass("done");
      }});
```

data

类型：String

发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称。如 {foo:["bar1", "bar2"]} 转换为 '&foo=bar1&foo=bar2'。

dataFilter

类型：Function

给 Ajax 返回的原始数据的进行预处理的函数。提供 data 和 type 两个参数：data 是 Ajax 返回的原始数据，type 是调用 jQuery.ajax 时提供的 dataType 参数。函数返回的值将由 jQuery 进一步处理。

dataType

类型：String

预期服务器返回的数据类型。如果不指定，jQuery 将自动根据 HTTP 包 MIME 信息来智能判断，比如 XML MIME 类型就被识别为 XML。在 1.4 中，JSON 就会生成一个 JavaScript 对象，而 script 则会执行这个脚本。随后服务器端返回的数据会根据这个值解析后，传递给回调函数。可用值:

- "xml": 返回 XML 文档，可用 jQuery 处理。
- "html": 返回纯文本 HTML 信息；包含的 script 标签会在插入 dom 时执行。
- "script": 返回纯文本 JavaScript 代码。不会自动缓存结果。除非设置了 "cache" 参数。注意：在远程请求时(不在同一个域下)，所有 POST 请求都将转为 GET 请求。（因为将使用 DOM 的 script标签来加载）
- "json": 返回 JSON 数据 。
- "jsonp": JSONP 格式。使用 JSONP 形式调用函数时，如 "myurl?callback=?" jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。
- "text": 返回纯文本字符串

error

类型：Function

默认值: 自动判断 (xml 或 html)。请求失败时调用此函数。

有以下三个参数：XMLHttpRequest 对象、错误信息、（可选）捕获的异常对象。

如果发生了错误，错误信息（第二个参数）除了得到 null 之外，还可能是 "timeout", "error", "notmodified" 和 "parsererror"。

这是一个 Ajax 事件。

global

类型：Boolean

是否触发全局 AJAX 事件。默认值: true。设置为 false 将不会触发全局 AJAX 事件，如 ajaxStart 或 ajaxStop 可用于控制不同的 Ajax 事件。

ifModified

类型：Boolean

仅在服务器数据改变时获取新数据。默认值: false。使用 HTTP 包 Last-Modified 头信息判断。在 jQuery 1.4 中，它也会检查服务器指定的 'etag' 来确定数据没有被修改过。

jsonp

类型：String

在一个 jsonp 请求中重写回调函数的名字。这个值用来替代在 "callback=?" 这种 GET 或 POST 请求中 URL 参数里的 "callback" 部分，比如 {jsonp:'onJsonPLoad'} 会导致将 "onJsonPLoad=?" 传给服务器。

jsonpCallback

类型：String

为 jsonp 请求指定一个回调函数名。这个值将用来取代 jQuery 自动生成的随机函数名。这主要用来让 jQuery 生成度独特的函数名，这样管理请求更容易，也能方便地提供回调函数和错误处理。你也可以在想让浏览器缓存 GET 请求的时候，指定这个回调函数名。

password

类型：String

用于响应 HTTP 访问认证请求的密码

processData

类型：Boolean

默认值: true。默认情况下，通过data选项传递进来的数据，如果是一个对象(技术上讲只要不是字符串)，都会处理转化成一个查询字符串，以配合默认内容类型 "application/x-www-form-urlencoded"。如果要发送 DOM 树信息或其它不希望转换的信息，请设置为 false。

scriptCharset

类型：String

只有当请求时 dataType 为 "jsonp" 或 "script"，并且 type 是 "GET" 才会用于强制修改 charset。通常只在本地和远程的内容编码不同时使用。

success

类型：Function

请求成功后的回调函数。

参数：由服务器返回，并根据 dataType 参数进行处理后的数据；描述状态的字符串。

这是一个 Ajax 事件。

traditional

类型：Boolean

如果你想要用传统的方式来序列化数据，那么就设置为 true。请参考工具分类下面的 jQuery.param 方法。

timeout

类型：Number

设置请求超时时间（毫秒）。此设置将覆盖全局设置。

type

类型：String

默认值: "GET")。请求方式 ("POST" 或 "GET")， 默认为 "GET"。注意：其它 HTTP 请求方法，如 PUT 和 DELETE 也可以使用，但仅部分浏览器支持。

url

类型：String

默认值: 当前页地址。发送请求的地址。

username

类型：String

用于响应 HTTP 访问认证请求的用户名。

xhr

类型：Function

需要返回一个 XMLHttpRequest 对象。默认在 IE 下是 ActiveXObject 而其他情况下是 XMLHttpRequest 。用于重写或者提供一个增强的 XMLHttpRequest 对象。这个参数在 jQuery 1.3 以前不可用。

回调函数

如果要处理 $.ajax() 得到的数据，则需要使用回调函数：beforeSend、error、dataFilter、success、complete。

beforeSend

在发送请求之前调用，并且传入一个 XMLHttpRequest 作为参数。

error

在请求出错时调用。传入 XMLHttpRequest 对象，描述错误类型的字符串以及一个异常对象（如果有的话）

dataFilter

在请求成功之后调用。传入返回的数据以及 "dataType" 参数的值。并且必须返回新的数据（可能是处理过的）传递给 success 回调函数。

success

当请求之后调用。传入返回后的数据，以及包含成功代码的字符串。

complete

当请求完成之后调用这个函数，无论成功或失败。传入 XMLHttpRequest 对象，以及一个包含成功或错误代码的字符串。

数据类型

$.ajax() 函数依赖服务器提供的信息来处理返回的数据。如果服务器报告说返回的数据是 XML，那么返回的结果就可以用普通的 XML 方法或者 jQuery 的选择器来遍历。如果见得到其他类型，比如 HTML，则数据就以文本形式来对待。

通过 dataType 选项还可以指定其他不同数据处理方式。除了单纯的 XML，还可以指定 html、json、jsonp、script 或者 text。

其中，text 和 xml 类型返回的数据不会经过处理。数据仅仅简单的将 XMLHttpRequest 的 responseText 或 responseHTML 属性传递给 success 回调函数。

注意：我们必须确保网页服务器报告的 MIME 类型与我们选择的 dataType 所匹配。比如说，XML的话，服务器端就必须声明 text/xml 或者 application/xml 来获得一致的结果。

如果指定为 html 类型，任何内嵌的 JavaScript 都会在 HTML 作为一个字符串返回之前执行。类似地，指定 script 类型的话，也会先执行服务器端生成 JavaScript，然后再把脚本作为一个文本数据返回。

如果指定为 json 类型，则会把获取到的数据作为一个 JavaScript 对象来解析，并且把构建好的对象作为结果返回。为了实现这个目的，它首先尝试使用 JSON.parse()。如果浏览器不支持，则使用一个函数来构建。

JSON 数据是一种能很方便通过 JavaScript 解析的结构化数据。如果获取的数据文件存放在远程服务器上（域名不同，也就是跨域获取数据），则需要使用 jsonp 类型。使用这种类型的话，会创建一个查询字符串参数 callback=? ，这个参数会加在请求的 URL 后面。服务器端应当在 JSON 数据前加上回调函数名，以便完成一个有效的 JSONP 请求。如果要指定回调函数的参数名来取代默认的 callback，可以通过设置 $.ajax() 的 jsonp 参数。

注意：JSONP 是 JSON 格式的扩展。它要求一些服务器端的代码来检测并处理查询字符串参数。

如果指定了 script 或者 jsonp 类型，那么当从服务器接收到数据时，实际上是用了 <script> 标签而不是 XMLHttpRequest 对象。这种情况下，$.ajax() 不再返回一个 XMLHttpRequest 对象，并且也不会传递事件处理函数，比如 beforeSend。

发送数据到服务器

默认情况下，Ajax 请求使用 GET 方法。如果要使用 POST 方法，可以设定 type 参数值。这个选项也会影响 data 选项中的内容如何发送到服务器。

data 选项既可以包含一个查询字符串，比如 key1=value1&key2=value2 ，也可以是一个映射，比如 {key1: 'value1', key2: 'value2'} 。如果使用了后者的形式，则数据再发送器会被转换成查询字符串。这个处理过程也可以通过设置 processData 选项为 false 来回避。如果我们希望发送一个 XML 对象给服务器时，这种处理可能并不合适。并且在这种情况下，我们也应当改变 contentType 选项的值，用其他合适的 MIME 类型来取代默认的 application/x-www-form-urlencoded 。

高级选项

global 选项用于阻止响应注册的回调函数，比如 .ajaxSend，或者 ajaxError，以及类似的方法。这在有些时候很有用，比如发送的请求非常频繁且简短的时候，就可以在 ajaxSend 里禁用这个。

如果服务器需要 HTTP 认证，可以使用用户名和密码可以通过 username 和 password 选项来设置。

Ajax 请求是限时的，所以错误警告被捕获并处理后，可以用来提升用户体验。请求超时这个参数通常就保留其默认值，要不就通过 jQuery.ajaxSetup 来全局设定，很少为特定的请求重新设置 timeout 选项。

默认情况下，请求总会被发出去，但浏览器有可能从它的缓存中调取数据。要禁止使用缓存的结果，可以设置 cache 参数为 false。如果希望判断数据自从上次请求后没有更改过就报告出错的话，可以设置 ifModified 为 true。

scriptCharset 允许给 <script> 标签的请求设定一个特定的字符集，用于 script 或者 jsonp 类似的数据。当脚本和页面字符集不同时，这特别好用。

Ajax 的第一个字母是 asynchronous 的开头字母，这意味着所有的操作都是并行的，完成的顺序没有前后关系。$.ajax() 的 async 参数总是设置成true，这标志着在请求开始后，其他代码依然能够执行。强烈不建议把这个选项设置成 false，这意味着所有的请求都不再是异步的了，这也会导致浏览器被锁死。

$.ajax 函数返回它创建的 XMLHttpRequest 对象。通常 jQuery 只在内部处理并创建这个对象，但用户也可以通过 xhr 选项来传递一个自己创建的 xhr 对象。返回的对象通常已经被丢弃了，但依然提供一个底层接口来观察和操控请求。比如说，调用对象上的 .abort() 可以在请求完成前挂起请求。

jQuery ajax - ajaxComplete() 方法

实例

当 AJAX 请求正在进行时显示“正在加载”的指示：

```
$("#txt").ajaxStart(function(){
  $("#wait").css("display","block");
});
$("#txt").ajaxComplete(function(){
  $("#wait").css("display","none");
});
```

定义和用法

ajaxComplete() 方法在 AJAX 请求完成时执行函数。它是一个 Ajax 事件。

与 ajaxSuccess() 不同，通过 ajaxComplete() 方法规定的函数会在请求完成时运行，即使请求并未成功。

语法

```
.jQueryajaxComplete(_function(event,xhr,options)_)
```

参数	描述
function(event,xhr,options)	必需。规定当请求完成时运行的函数。额外的参数： *event* - 包含 event 对象 *xhr* - 包含 XMLHttpRequest 对象 *options* - 包含 AJAX 请求中使用的选项

详细说明

XMLHttpRequest 对象和设置作为参数传递给回调函数。

jQuery ajax - ajaxError() 方法

实例

当 AJAX 请求失败时，触发提示框：

```
$("div").ajaxError(function(){
  alert("An error occurred!");
});
```

定义和用法

ajaxError() 方法在 AJAX 请求发生错误时执行函数。它是一个 Ajax 事件。

语法

```
.ajaxError(_function(event,xhr,options,exc)_)
```

参数	描述
function(event,xhr,options,exc)	必需。规定当请求失败时运行的函数。额外的参数： *event* - 包含 event 对象 *xhr* - 包含 XMLHttpRequest 对象 *options* - 包含 AJAX 请求中使用的选项 *exc* - 包含 JavaScript exception

详细说明

XMLHttpRequest 对象和设置作为参数传递给回调函数。捕捉到的错误可作为最后一个参数传递：

```
function (event, XMLHttpRequest, ajaxOptions, thrownError) {
 // thrownError 只有当异常发生时才会被传递 this;
 }
```

亲自试一试 - 实例

使用 *xhr* 和 *options* 参数

如何使用 *options* 参数来获得更有用的错误消息。

jQuery ajax - ajaxSend() 方法

实例

当 AJAX 请求即将发送时，改变 div 元素的内容：

```
$("div").ajaxSend(function(e,xhr,opt){
  $(this).html("Requesting " + opt.url);
});
```

定义和用法

ajaxSend() 方法在 AJAX 请求开始时执行函数。它是一个 Ajax 事件。

语法

```
.ajaxSend([_function(event,xhr,options)_])
```

参数	描述
function(event,xhr,options)	必需。规定当请求开始时执行函数。额外的参数： *event* - 包含 event 对象 *xhr* - 包含 XMLHttpRequest 对象 *options* - 包含 AJAX 请求中使用的选项

详细说明

XMLHttpRequest 对象和设置作为参数传递给回调函数。

jQuery ajax - ajaxSetup() 方法

实例

为所有 AJAX 请求设置默认 URL 和 success 函数：

```
$("button").click(function(){
  $.ajaxSetup({url:"demo_ajax_load.txt",success:function(result){
    $("div").html(result);}});
  $.ajax();
});
```

定义和用法

jQuery.ajaxSetup() 方法设置全局 AJAX 默认选项。

语法

```
jQuery.ajaxSetup(_name:value_, _name:value_, ...)
```

示例

设置 AJAX 请求默认地址为 "/xmlhttp/"，禁止触发全局 AJAX 事件，用 POST 代替默认 GET 方法。其后的 AJAX 请求不再设置任何选项参数：

```
$.ajaxSetup({
  url: "/xmlhttp/",
  global: false,
  type: "POST"
});
$.ajax({ data: myData });
```

参数	描述
name:value	可选。使用名称/值对来规定 AJAX 请求的设置。

注释：参数见 '$.ajax' 说明。

jQuery ajax - ajaxStart() 方法

实例

当 AJAX 请求开始时，显示“加载中”的指示：

```
$("div").ajaxStart(function(){
  $(this).html("<img src='demo_wait.gif' />");
});
```

定义和用法

ajaxStart() 方法在 AJAX 请求发送前执行函数。它是一个 Ajax 事件。

详细说明

无论在何时发送 Ajax 请求，jQuery 都会检查是否存在其他 Ajax 请求。如果不存在，则 jQuery 会触发该 ajaxStart 事件。在此时，由 .ajaxStart() 方法注册的任何函数都会被执行。

语法

```
.ajaxStart(_function()_)
```

参数	描述
function()	规定当 AJAX 请求开始时运行的函数。

示例

AJAX 请求开始时显示信息：

```
$("#loading").ajaxStart(function(){
  $(this).show();
});
```

jQuery ajax - ajaxStop() 方法

实例

当所有 AJAX 请求完成时，触发一个提示框：

```
$("div").ajaxStop(function(){
  alert("所有 AJAX 请求已完成");
});
```

定义和用法

ajaxStop() 方法在 AJAX 请求结束时执行函数。它是一个 Ajax 事件。

详细说明

无论 Ajax 请求在何时完成，jQuery 都会检查是否存在其他 Ajax 请求。如果不存在，则 jQuery 会触发该 ajaxStop 事件。在此时，由 .ajaxStop() 方法注册的任何函数都会被执行。

语法

```
.ajaxStop(_function()_)
```

参数	描述
function()	规定当 AJAX 请求完成时运行的函数。

示例

AJAX 请求结束后隐藏信息：

```
$("#loading").ajaxStop(function(){
  $(this).hide();
});
```

jQuery ajax - ajaxSuccess() 方法

实例

当 AJAX 请求成功完成时，触发提示框：

```
$("div").ajaxSuccess(function(){
  alert("AJAX 请求已成功完成");
});
```

定义和用法

ajaxSuccess() 方法在 AJAX 请求成功时执行函数。它是一个 Ajax 事件。

详细说明

XMLHttpRequest 对象和设置作为参数传递给回调函数。

无论 Ajax 请求在何时成功完成，jQuery 都会触发该 ajaxSuccess 事件。在此时，由 .ajaxSuccess() 方法注册的任何函数都会被执行。

语法

```
.ajaxSuccess(_function(event,xhr,options)_)
```

参数	描述
function(event,xhr,options)	必需。规定当请求成功时运行的函数。额外的参数： *event* - 包含 event 对象 *xhr* - 包含 XMLHttpRequest 对象 *options* - 包含 AJAX 请求中使用的选项

示例

AJAX 请求成功后显示消息：

```
$("#msg").ajaxSuccess(function(evt, request, settings){
  $(this).append("<p>请求成功!</p>");
});
```

jQuery ajax - get() 方法

实例

使用 AJAX 的 GET 请求来改变 div 元素的文本：

```
$("button").click(function(){
  $.get("demo_ajax_load.txt", function(result){
    $("div").html(result);
  });
});
```

定义和用法

get() 方法通过远程 HTTP GET 请求载入信息。

这是一个简单的 GET 请求功能以取代复杂 $.ajax 。请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 $.ajax。

语法

```
$(_selector_).get(_url_,_data_,_success(response,status,xhr)_,_data
```

参数	描述
url	必需。规定将请求发送的哪个 URL。
data	可选。规定连同请求发送到服务器的数据。
success(response,status,xhr)	可选。规定当请求成功时运行的函数。额外的参数： response - 包含来自请求的结果数据 status - 包含请求的状态 xhr - 包含 XMLHttpRequest 对象
dataType	可选。规定预计的服务器响应的数据类型。默认地，jQuery 将智能判断。可能的类型： "xml" "html" "text" "script" "json" "jsonp"

详细说明

该函数是简写的 Ajax 函数，等价于：

```
$.ajax({
  url: _url_,
  data: _data_,
  success: _success_,
  dataType: _dataType_
});
```

根据响应的不同的 MIME 类型，传递给 success 回调函数的返回数据也有所不同，这些数据可以是 XML root 元素、文本字符串、JavaScript 文件或者 JSON 对象。也可向 success 回调函数传递响应的文本状态。

对于 jQuery 1.4，也可以向 success 回调函数传递 XMLHttpRequest 对象。

示例

请求 test.php 网页，忽略返回值：

```
$.get("test.php");
```

更多示例

例子 1

请求 test.php 网页，传送2个参数，忽略返回值：

```
$.get("test.php", { name: "John", time: "2pm" } );
```

例子 2

显示 test.php 返回值(HTML 或 XML，取决于返回值)：

```
$.get("test.php", function(data){
  alert("Data Loaded: " + data);
});
```

例子 3

显示 test.cgi 返回值(HTML 或 XML，取决于返回值)，添加一组请求参数：

```
$.get("test.cgi", { name: "John", time: "2pm" },
  function(data){
    alert("Data Loaded: " + data);
  });
```

jQuery ajax - getJSON() 方法

实例

使用 AJAX 请求来获得 JSON 数据，并输出结果：

```
$("button").click(function(){
  $.getJSON("demo_ajax_json.js",function(result){
    $.each(result, function(i, field){
      $("div").append(field + " ");
    });
  });
});
```

定义和用法

通过 HTTP GET 请求载入 JSON 数据。

在 jQuery 1.2 中，您可以通过使用 JSONP 形式的回调函数来加载其他网域的 JSON 数据，如 "myurl?callback=?"。jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。 注意：此行以后的代码将在这个回调函数执行前执行。

语法

```
jQuery.getJSON(_url_,_data_,_success(data,status,xhr)_)
```

参数	描述
url	必需。规定将请求发送的哪个 URL。
data	可选。规定连同请求发送到服务器的数据。
success(data,status,xhr)	可选。规定当请求成功时运行的函数。额外的参数： *response* - 包含来自请求的结果数据 *status* - 包含请求的状态 *xhr* - 包含 XMLHttpRequest 对象

详细说明

该函数是简写的 Ajax 函数，等价于：

```
$.ajax({
  url: _url_,
  data: _data_,
  success: _callback_,
  dataType: json
});
```

发送到服务器的数据可作为查询字符串附加到 URL 之后。如果 *data* 参数的值是对象（映射），那么在附加到 URL 之前将转换为字符串，并进行 URL 编码。

传递给 *callback* 的返回数据，可以是 JavaScript 对象，或以 JSON 结构定义的数组，并使用 $.parseJSON() 方法进行解析。

示例

从 test.js 载入 JSON 数据并显示 JSON 数据中一个 name 字段数据：

```
$.getJSON("test.js", function(json){
  alert("JSON Data: " + json.users[3].name);
});
```

更多示例

例子 1

从 Flickr JSONP API 载入 4 张最新的关于猫的图片：

HTML 代码：

```
<div id="images"></div>
```

jQuery 代码：

```
$.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?
tags=cat&tagmode=any&format=json&jsoncallback=?", function(data){
  $.each(data.items, function(i,item){
    $("<img/>").attr("src", item.media.m).appendTo("#images");
    if ( i == 3 ) return false;
  });
});
```

例子 2

从 test.js 载入 JSON 数据，附加参数，显示 JSON 数据中一个 name 字段数据：

```
$.getJSON("test.js", { name: "John", time: "2pm" }, function(json)·
  alert("JSON Data: " + json.users[3].name);
});
```

jQuery ajax - getScript() 方法

实例

通过 AJAX 请求来获得并运行一个 JavaScript 文件：

```
$("button").click(function(){
  $.getScript("demo_ajax_script.js");
});
```

定义和用法

getScript() 方法通过 HTTP GET 请求载入并执行 JavaScript 文件。

语法

```
jQuery.getScript(_url_,_success(response,status)_)
```

参数	描述
url	将要请求的 URL 字符串。
success(response,status)	可选。规定请求成功后执行的回调函数。额外的参数： *response* - 包含来自请求的结果数据 *status* - 包含请求的状态（"success", "notmodified", "error", "timeout" 或 "parsererror"）

详细说明

该函数是简写的 Ajax 函数，等价于：

```
$.ajax({
  url: _url_,
  dataType: "script",
  success: _success_
});
```

这里的回调函数会传入返回的 JavaScript 文件。这通常不怎么有用，因为那时脚本已经运行了。

载入的脚本在全局环境中执行，因此能够引用其他变量，并使用 jQuery 函数。

比如加载一个 test.js 文件，里边包含下面这段代码：

```
$(".result").html("<p>Lorem ipsum dolor sit amet.</p>");
```

通过引用该文件名，就可以载入并运行这段脚本：

```
$.getScript("ajax/test.js", function() {
  alert("Load was performed.");
});
```

注释：jQuery 1.2 版本之前，getScript 只能调用同域 JS 文件。 1.2中，您可以跨域调用 JavaScript 文件。注意：Safari 2 或更早的版本不能在全局作用域中同步执行脚本。如果通过 getScript 加入脚本，请加入延时函数。

更多实例

例子 1

加载并执行 test.js：

```
$.getScript("test.js");
```

例子 2

加载并执行 test.js，成功后显示信息：

```
$.getScript("test.js", function(){
  alert("Script loaded and executed.");
});
```

例子 3

载入 jQuery 官方颜色动画插件 成功后绑定颜色变化动画：

HTML 代码：

```
<button id="go">Run</button>
<div class="block"></div>
```

jQuery 代码：

```
jQuery.getScript("http://dev.jquery.com/view/trunk/plugins/color/jq
 function(){
  $("#go").click(function(){
    $(".block").animate( { backgroundColor: 'pink' }, 1000)
      .animate( { backgroundColor: 'blue' }, 1000);
  });
});
```

jQuery ajax - load() 方法

实例

使用 AJAX 请求来改变 div 元素的文本：

```
$("button").click(function(){
  $("div").load('demo_ajax_load.txt');
});
```

您可以在页面底部找到更多 TIY 实例

定义和用法

load() 方法通过 AJAX 请求从服务器加载数据，并把返回的数据放置到指定的元素中。

注释：还存在一个名为 load 的 jQuery 事件方法。调用哪个，取决于参数。

语法

```
load(_url_,_data_,_function(response,status,xhr)_)
```

参数	描述
url	规定要将请求发送到哪个 URL。
data	可选。规定连同请求发送到服务器的数据。
function(response,status,xhr)	可选。规定当请求完成时运行的函数。额外的参数： *response* - 包含来自请求的结果数据 *status* - 包含请求的状态（"success", "notmodified", "error", "timeout" 或 "parsererror"） *xhr* - 包含 XMLHttpRequest 对象

详细说明

该方法是最简单的从服务器获取数据的方法。它几乎与 $.get(url, data, success) 等价，不同的是它不是全局函数，并且它拥有隐式的回调函数。当侦测到成功的响应时（比如，当 textStatus 为 "success" 或 "notmodified" 时），.load() 将匹配元素

的 HTML 内容设置为返回的数据。这意味着该方法的大多数使用会非常简单：

```
$("#result").load("ajax/test.html");
```

如果提供回调函数，则会在执行 post-processing 之后执行该函数：

```
$("#result").load("ajax/test.html", function() {
  alert("Load was performed.");
});
```

上面的两个例子中，如果当前文档不包含 "result" ID，则不会执行 .load() 方法。

如果提供的数据是对象，则使用 POST 方法；否则使用 GET 方法。

加载页面片段

.load() 方法，与 $.get() 不同，允许我们规定要插入的远程文档的某个部分。这一点是通过 url 参数的特殊语法实现的。如果该字符串中包含一个或多个空格，紧接第一个空格的字符串则是决定所加载内容的 jQuery 选择器。

我们可以修改上面的例子，这样就可以使用所获得文档的某部分：

```
$("#result").load("ajax/test.html #container");
```

如果执行该方法，则会取回 ajax/test.html 的内容，不过然后，jQuery 会解析被返回的文档，来查找带有容器 ID 的元素。该元素，连同其内容，会被插入带有结果 ID 的元素中，所取回文档的其余部分会被丢弃。

jQuery 使用浏览器的 .innerHTML 属性来解析被取回的文档，并把它插入当前文档。在此过程中，浏览器常会从文档中过滤掉元素，比如 <html>, <title> 或 <head> 元素。结果是，由 .load() 取回的元素可能与由浏览器直接取回的文档不完全相同。

注释：由于浏览器安全方面的限制，大多数 "Ajax" 请求遵守同源策略；请求无法从不同的域、子域或协议成功地取回数据。

更多实例

例子 1

加载 feeds.html 文件内容：

```
$("#feeds").load("feeds.html");
```

例子 2

与上面的实例类似，但是以 POST 形式发送附加参数并在成功时显示信息：

```
$("#feeds").load("feeds.php", {limit: 25}, function(){
  alert("The last 25 entries in the feed have been loaded");
});
```

例子 3

加载文章侧边栏导航部分至一个无序列表：

HTML 代码：

```
<b>jQuery Links:</b>
<ul id="links"></ul>
```

jQuery 代码：

```
$("#links").load("/Main_Page #p-Getting-Started li");
```

更多 TIY 实例

生成 AJAX 请求，并通过该请求发送数据

如何使用 data 参数通过 AJAX 请求来发送数据。（本例在 AJAX 教程中解释过。）

生成 AJAX 请求，并使用回调函数

如何使用 function 参数处理来自 AJAX 请求的数据结果。

生成带有错误的 AJAX 请求

如何使用 function 参数来处理 AJAX 请求中的错误（使用 XMLHttpRequest 参数）。

jQuery ajax - param() 方法

实例

序列化一个 *key/value* 对象：

```
var params = { width:1900, height:1200 };
var str = jQuery.param(params);
$("#results").text(str);
```

结果:

```
width=1680&height=1050
```

TIY 实例

输出序列化对象的结果：

```
$("button").click(function(){
  $("div").text($.param(personObj));
});
```

定义和用法

param() 方法创建数组或对象的序列化表示。

该序列化值可在进行 AJAX 请求时在 URL 查询字符串中使用。

语法

```
jQuery.param(_object_,_traditional_)
```

参数	描述
object	要进行序列化的数组或对象。
traditional	规定是否使用传统的方式浅层进行序列化（参数序列化）。

详细说明

param() 方法用于在内部将元素值转换为序列化的字符串表示。请参阅 .serialize() 了解更多信息。

对于 jQuery 1.3，如果传递的参数是一个函数，那么用 .param() 会得到这个函数的返回值，而不是把这个函数作为一个字符串来返回。

对于 jQuery 1.4，.param() 方法将会通过深度递归的方式序列化对象，以便符合现代化脚本语言的需求，比如 PHP、Ruby on Rails 等。你可以通过设置 jQuery.ajaxSettings.traditional = true; 来全局地禁用这个功能。

如果被传递的对象在数组中，则必须是以 .serializeArray() 的返回值为格式的对象数组：

```
[{name:"first",value:"Rick"},
{name:"last",value:"Astley"},
{name:"job",value:"Rock Star"}]
```

注意：因为有些框架在解析序列化数字的时候能力有限，所以当传递一些含有对象或嵌套数组的数组作为参数时，请务必小心！

在 jQuery 1.4 中，HTML5 的 input 元素也会被序列化。

更多实例

我们可以如下显示对象的查询字符串表示以及 URI 编码版本：

```
var myObject = {
  a: {
    one: 1,
    two: 2,
    three: 3
  },
  b: [1,2,3]
};
var recursiveEncoded = $.param(myObject);
var recursiveDecoded = decodeURIComponent($.param(myObject));

alert(recursiveEncoded);
alert(recursiveDecoded);
```

recursiveEncoded 和 recursiveDecoded 的值输出如下：

```
a%5Bone%5D=1&a%5Btwo%5D=2&a%5Bthree%5D=3&b%5B%5D=1&b%5B%5D=2&b%5B%5
a[one]=1&a[two]=2&a[three]=3&b[]=1&b[]=2&b[]=3
```

可以将 traditional 参数设置为 true，来模拟 jQuery 1.4 之前版本中 $.param() 的行为：

```
var myObject = {
  a: {
    one: 1,
    two: 2,
    three: 3
  },
  b: [1,2,3]
};
var shallowEncoded = $.param(myObject, true);
var shallowDecoded = decodeURIComponent(shallowEncoded);

alert(shallowEncoded);
alert(shallowDecoded);
```

recursiveEncoded 和 recursiveDecoded 的值输出如下：

```
a=%5Bobject+Object%5D&b=1&b=2&b=3
a=[object+Object]&b=1&b=2&b=3
```

jQuery ajax - post() 方法

实例

请求 test.php 网页，忽略返回值：

```
$.post("test.php");
```

TIY 实例

通过 AJAX POST 请求改变 div 元素的文本：

```
$("input").keyup(function(){
  txt=$("input").val();
  $.post("demo_ajax_gethint.asp",{suggest:txt},function(result){
    $("span").html(result);
  });
});
```

定义和用法

post() 方法通过 HTTP POST 请求从服务器载入数据。

语法

```
jQuery.post(_url_,_data_,_success(data, textStatus, jqXHR)_,_dataTy
```

参数	描述
url	必需。规定把请求发送到哪个 URL。
data	可选。映射或字符串值。规定连同请求发送到服务器的数据。
success(data, textStatus, jqXHR)	可选。请求成功时执行的回调函数。
dataType	可选。规定预期的服务器响应的数据类型。默认执行智能判断（xml、json、script 或 html）。

详细说明

该函数是简写的 Ajax 函数，等价于：

```
$.ajax({
  type: 'POST',
  url: _url_,
  data: _data_,
  success: _success_,
  dataType: _dataType_
});
```

根据响应的不同的 MIME 类型，传递给 success 回调函数的返回数据也有所不同，这些数据可以是 XML 根元素、文本字符串、JavaScript 文件或者 JSON 对象。也可向 success 回调函数传递响应的文本状态。

对于 jQuery 1.5，也可以向 success 回调函数传递 jqXHR 对象（jQuery 1.4 中传递的是 XMLHttpRequest 对象）。

大部分实现会规定一个 success 函数：

```
$.post("ajax/test.html", function(data) {
  $(".result").html(data);
});
```

本例读取被请求的 HTML 片段，并插入页面中。

通过 POST 读取的页面不被缓存，因此 jQuery.ajaxSetup() 中的 cache 和 ifModified 选项不会影响这些请求。

注释：由于浏览器安全方面的限制，大多数 "Ajax" 请求遵守同源策略；请求无法从不同的域、子域或协议成功地取回数据。

注释：如果由 jQuery.post() 发起的请求返回错误代码，那么不会有任何提示，除非脚本已调用了全局的 .ajaxError() 方法。或者对于 jQuery 1.5，jQuery.post() 返回的 jqXHR 对象的 .error() 方法也可以用于错误处理。

jqXHR 对象

对于 jQuery 1.5，所有 jQuery 的 AJAX 方法返回的是 XMLHTTPRequest 对象的超集。由 $.post() 返回的 jQuery XHR 对象或 "jqXHR,"实现了约定的接口，赋予其所有的属性、方法，以及约定的行为。出于对由 $.ajax() 使用的回调函数名称便利性和一致性的考虑，它提供了 .error(), .success() 以及 .complete() 方法。这些方法使用请求终止时调用的函数参数，该函数接受与对应命名的 $.ajax() 回调函数相同的参数。

jQuery 1.5 中的约定接口同样允许 jQuery 的 Ajax 方法，包括 $.post()，来链接同一请求的多个 .success()、.complete() 以及 .error() 回调函数，甚至会在请求也许已经完成后分配这些回调函数。

```
// 请求生成后立即分配处理程序，请记住该请求针对 jqxhr 对象
    var jqxhr = $.post("example.php", function() {
      alert("success");
    })
    .success(function() { alert("second success"); })
    .error(function() { alert("error"); })
    .complete(function() { alert("complete"); });

    // 在这里执行其他任务

    // 为上面的请求设置另一个完成函数
    jqxhr.complete(function(){ alert("second complete"); });
```

更多实例

例子 1

请求 test.php 页面，并一起发送一些额外的数据（同时仍然忽略返回值）：

```
$.post("test.php", { name: "John", time: "2pm" } );
```

例子 2

向服务器传递数据数组（同时仍然忽略返回值）：

```
$.post("test.php", { 'choices[]': ["Jon", "Susan"] });
```

例子 3

使用 ajax 请求发送表单数据：

```
$.post("test.php", $("#testform").serialize());
```

例子 4

输出来自请求页面 test.php 的结果（HTML 或 XML，取决于所返回的内容）：

```
$.post("test.php", function(data){
   alert("Data Loaded: " + data);
 });
```

例子 5

向页面 test.php 发送数据，并输出结果（HTML 或 XML，取决于所返回的内容）：

```
$.post("test.php", { name: "John", time: "2pm" },
   function(data){
     alert("Data Loaded: " + data);
   });
```

例子 6

获得 test.php 页面的内容，并存储为 XMLHttpResponse 对象，并通过 process() 这个 JavaScript 函数进行处理：

```
$.post("test.php", { name: "John", time: "2pm" },
   function(data){
     process(data);
   }, "xml");
```

例子 7

获得 test.php 页面返回的 json 格式的内容：

```
$.post("test.php", { "func": "getNameAndTime" },
   function(data){
     alert(data.name); // John
     console.log(data.time); //  2pm
   }, "json");
```

jQuery ajax - serialize() 方法

实例

输出序列化表单值的结果：

```
$("button").click(function(){
  $("div").text($("form").serialize());
});
```

定义和用法

serialize() 方法通过序列化表单值，创建 URL 编码文本字符串。

您可以选择一个或多个表单元素（比如 input 及/或 文本框），或者 form 元素本身。

序列化的值可在生成 AJAX 请求时用于 URL 查询字符串中。

语法

```
$(_selector_).serialize()
```

详细说明

.serialize() 方法创建以标准 URL 编码表示的文本字符串。它的操作对象是代表表单元素集合的 jQuery 对象。

表单元素有几种类型：

```
<form>
  <div><input type="text" name="a" value="1" id="a" /></div>
  <div><input type="text" name="b" value="2" id="b" /></div>
  <div><input type="hidden" name="c" value="3" id="c" /></div>
  <div>
    <textarea name="d" rows="8" cols="40">4</textarea>
  </div>
  <div><select name="e">
    <option value="5" selected="selected">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
  </select></div>
  <div>
    <input type="checkbox" name="f" value="8" id="f" />
  </div>
  <div>
    <input type="submit" name="g" value="Submit" id="g" />
  </div>
</form>
```

.serialize() 方法可以操作已选取个别表单元素的 jQuery 对象，比如 <input>, <textarea> 以及 <select>。不过，选择 <form> 标签本身进行序列化一般更容易些：

```
$('form').submit(function() {
  alert($(this).serialize());
  return false;
});
```

输出标准的查询字符串：

```
a=1&b=2&c=3&d=4&e=5
```

注释：只会将"成功的控件"序列化为字符串。如果不使用按钮来提交表单，则不对提交按钮的值序列化。如果要表单元素的值包含到序列字符串中，元素必须使用 name 属性。

jQuery ajax - serializeArray() 方法

实例

输出以数组形式序列化表单值的结果：

```
$("button").click(function(){
  x=$("form").serializeArray();
  $.each(x, function(i, field){
    $("#results").append(field.name + ":" + field.value + " ");
  });
});
```

定义和用法

serializeArray() 方法通过序列化表单值来创建对象数组（名称和值）。

您可以选择一个或多个表单元素（比如 input 及/或 textarea），或者 form 元素本身。

语法

```
$(_selector_).serializeArray()
```

详细说明

serializeArray() 方法序列化表单元素（类似 .serialize() 方法），返回 JSON 数据结构数据。

注意：此方法返回的是 JSON 对象而非 JSON 字符串。需要使用插件或者第三方库进行字符串化操作。

返回的 JSON 对象是由一个对象数组组成的，其中每个对象包含一个或两个名值对 —— name 参数和 value 参数（如果 value 不为空的话）。举例来说：

```
[
  {name: 'firstname', value: 'Hello'},
  {name: 'lastname', value: 'World'},
  {name: 'alias'}, // 值为空
]
```

.serializeArray() 方法使用了 W3C 关于 successful controls（有效控件） 的标准来检测哪些元素应当包括在内。特别说明，元素不能被禁用（禁用的元素不会被包括在内），并且元素应当有含有 name 属性。提交按钮的值也不会被序列化。文件选择元素的数据也不会被序列化。

该方法可以对已选择单独表单元素的对象进行操作，比如 <input>, <textarea>, 和 <select>。不过，更方便的方法是，直接选择 <form> 标签自身来进行序列化操作。

```
$("form").submit(function() {
  console.log($(this).serializeArray());
  return false;
});
```

上面的代码产生下面的数据结构（假设浏览器支持 console.log）：

```
[
  {
    name: a
    value: 1
  },
  {
    name: b
    value: 2
  },
  {
    name: c
    value: 3
  },
  {
    name: d
    value: 4
  },
  {
    name: e
    value: 5
  }
]
```

示例

取得表单内容并插入到网页中：

HTML 代码：

```
<p id="results"><b>Results:</b> </p>
<form>
  <select name="single">
    <option>Single</option>
    <option>Single2</option>
  </select>
  <select name="multiple" multiple="multiple">
    <option selected="selected">Multiple</option>
    <option>Multiple2</option>
    <option selected="selected">Multiple3</option>
  </select><br/>
  <input type="checkbox" name="check" value="check1"/> check1
  <input type="checkbox" name="check" value="check2" checked="checl
  <input type="radio" name="radio" value="radio1" checked="checked'
  <input type="radio" name="radio" value="radio2"/> radio2
</form>
```

jQuery 代码：

```
var fields = $("select, :radio").serializeArray();
jQuery.each( fields, function(i, field){
  $("#results").append(field.value + " ");
});
```

jQuery 参考手册 - 遍历

jQuery 遍历函数

jQuery 遍历函数包括了用于筛选、查找和串联元素的方法。

函数	描述
.add()	将元素添加到匹配元素的集合中。
.andSelf()	把堆栈中之前的元素集添加到当前集合中。
.children()	获得匹配元素集合中每个元素的所有子元素。
.closest()	从元素本身开始，逐级向上级元素匹配，并返回最先匹配的祖先元素。
.contents()	获得匹配元素集合中每个元素的子元素，包括文本和注释节点。
.each()	对 jQuery 对象进行迭代，为每个匹配元素执行函数。
.end()	结束当前链中最近的一次筛选操作，并将匹配元素集合返回到前一次的状态。
.eq()	将匹配元素集合缩减为位于指定索引的新元素。
.filter()	将匹配元素集合缩减为匹配选择器或匹配函数返回值的新元素。
.find()	获得当前匹配元素集合中每个元素的后代，由选择器进行筛选。
.first()	将匹配元素集合缩减为集合中的第一个元素。
.has()	将匹配元素集合缩减为包含特定元素的后代的集合。
.is()	根据选择器检查当前匹配元素集合，如果存在至少一个匹配元素，则返回 true。
.last()	将匹配元素集合缩减为集合中的最后一个元素。
.map()	把当前匹配集合中的每个元素传递给函数，产生包含返回值的新 jQuery 对象。
.next()	获得匹配元素集合中每个元素紧邻的同辈元素。
.nextAll()	获得匹配元素集合中每个元素之后的所有同辈元素，由选择器进行筛选（可选）。
	获得每个元素之后所有的同辈元素，直到遇到匹配选择器的元

	素为止。
.not()	从匹配元素集合中删除元素。
.offsetParent()	获得用于定位的第一个父元素。
.parent()	获得当前匹配元素集合中每个元素的父元素，由选择器筛选（可选）。
.parents()	获得当前匹配元素集合中每个元素的祖先元素，由选择器筛选（可选）。
.parentsUntil()	获得当前匹配元素集合中每个元素的祖先元素，直到遇到匹配选择器的元素为止。
.prev()	获得匹配元素集合中每个元素紧邻的前一个同辈元素，由选择器筛选（可选）。
.prevAll()	获得匹配元素集合中每个元素之前的所有同辈元素，由选择器进行筛选（可选）。
.prevUntil()	获得每个元素之前所有的同辈元素，直到遇到匹配选择器的元素为止。
.siblings()	获得匹配元素集合中所有元素的同辈元素，由选择器筛选（可选）。
.slice()	将匹配元素集合缩减为指定范围的子集。

jQuery 遍历 - add() 方法

实例

找到所有 div 并添加边框。然后将所有段落添加到该 jQuery 对象，并把它们的背景设置为黄色：

```
$("div").css("border", "2px solid red")
        .add("p")
        .css("background", "yellow");
```

定义和用法

add() 方法将元素添加到匹配元素的集合中。

语法 1

```
.add(_selector_)
```

参数	描述
selector	字符串值，表示查找供添加到匹配元素集合的元素的选择器表达式。

语法 2

```
.add(_elements_)
```

参数	描述
elements	添加到匹配元素集合的一个或多个元素。

语法 3

```
.add(_html_)
```

参数	描述
html	添加到匹配元素集合的 HTML 片段。

语法 4

```
.add(_jQueryObject_)
```

参数	描述
jQueryObject	添加到匹配元素集合的已有 jQuery 对象。

语法 5

```
.add(_selector_, _context_)
```

参数	描述
selector	字符串值，表示查找供添加到匹配元素集合的元素的选择器表达式。
context	选择器开始进行匹配的位置。

详细说明

暂无。

jQuery 遍历 - andSelf() 方法

实例

找到所有 div，以及其中的所有段落，并为它们添加两个类名。请注意，由于未使用 .andSelf()，div 没有黄色背景色。

```
$("div").find("p").andSelf().addClass("border");
$("div").find("p").addClass("background");
```

定义和用法

add() 方法把堆栈中之前的元素集添加到当前集合。

语法

```
.andSelf()
```

详细说明

请思考这个拥有简单列表的页面：

```
<ul>
   <li>list item 1</li>
   <li>list item 2</li>
   <li class="third-item">list item 3</li>
   <li>list item 4</li>
   <li>list item 5</li>
</ul>
```

以下代码的结果是项目 3,4,5 拥有红色背景：

```
$("li.third-item").nextAll().andSelf()
  .css("background-color", "red");
```

首先，初始的选择器会定位项目 3，初始化的堆栈存有仅包含该项目的集合。调用 .nextAll() 会将项目 4, 5 的集合推入堆栈。最后，调用 .andSelf() 会合并这两个集合，所创建的 jQuery 对象指向按照文档顺序的所有三个项目：{[<li.third-item>,<li>,]}。

jQuery 遍历 - children() 方法

实例

找到类名为 "selected" 的所有 div 的子元素，并将其设置为蓝色：

```
$("div").children(".selected").css("color", "blue");
```

定义和用法

add() 方法返回匹配元素集合中每个元素的子元素，添加可选参数可通过选择器进行过滤。

语法

```
.children(_selector_)
```

参数	描述
selector	字符串值，包含匹配元素的选择器表达式。

详细说明

如果给定表示 DOM 元素集合的 jQuery 对象，.children() 方法允许我们检索 DOM 树中的这些元素，并用匹配元素构造新的 jQuery 对象。.find() 和 .children() 方法类似，不过后者只沿着 DOM 树向下遍历单一层级。

请注意，与大多数 jQuery 方法一样，.children() 不返回文本节点；如果需要获得包含文本和注释节点在内的所有子节点，请使用 .contents()。

该方法接受一个选择器表达式作为可选参数，与我们传递到 $() 的参数的类型是相同的。如果应用该选择器，将测试元素是否匹配该表达式，以此筛选这些元素。

请思考这个带有基础的嵌套列表的页面：

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

如果我们从 level-2 列表开始，我们可以找到它的子元素：

```
$('ul.level-2').children().css('background-color', 'red');
```

这行代码的结果是，项目 A, B, C 得到红色背景。由于我们没有应用选择器表达式，返回的 jQuery 对象包含了所有子元素。如果应用一个选择器的话，那么只会包括匹配的项目。

jQuery 遍历 - closest() 方法

实例

本例演示如何通过 closest() 完成事件委托。当被最接近的列表元素或其子后代元素被点击时，会切换黄色背景：

```
$( document ).bind("click", function( e ) {
    $( e.target ).closest("li").toggleClass("hilight");
  });
```

定义和用法

closest() 方法获得匹配选择器的第一个祖先元素，从当前元素开始沿 DOM 树向上。

语法

```
.closest(_selector_)
```

参数	描述
selector	字符串值，包含匹配元素的选择器表达式。

详细说明

如果给定表示 DOM 元素集合的 jQuery 对象，.closest() 方法允许我们检索 DOM 树中的这些元素以及它们的祖先元素，并用匹配元素构造新的 jQuery 对象。.parents() 和 .closest() 方法类似，它们都沿 DOM 树向上遍历。两者之间的差异尽管微妙，却很重要：

.closest()	.parents()
从当前元素开始	从父元素开始
沿 DOM 树向上遍历，直到找到已应用选择器的一个匹配为止。	沿 DOM 树向上遍历，直到文档的根元素为止，将每个祖先元素添加到一个临时的集合；如果应用了选择器，则会基于该选择器对这个集合进行筛选。
返回包含零个或一个元素的 jQuery 对象	返回包含零个、一个或多个元素的 jQuery 对象

请看下面的 HTML 片段：

```
<ul id="one" class="level-1">
  <li class="item-i">I</li>
  <li id="ii" class="item-ii">II
  <ul class="level-2">
    <li class="item-a">A</li>
    <li class="item-b">B
      <ul class="level-3">
        <li class="item-1">1</li>
        <li class="item-2">2</li>
        <li class="item-3">3</li>
      </ul>
    </li>
    <li class="item-c">C</li>
  </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

例子 1

假设我们执行一个从项目 A 开始的对 <ul> 元素的搜索：

```
$('li.item-a').closest('ul').css('background-color', 'red');
```

这会改变 level-2 <ul> 的颜色，这是因为当向上遍历 DOM 树时会第一个遇到该元素。

例子 2

假设我们搜索的是 <li> 元素：

```
$('li.item-a').closest('li').css('background-color', 'red');
```

这会改变列表项目 A 的颜色。在向上遍历 DOM 树之前，.closest() 方法会从 li 元素本身开始搜索，直到选择器匹配项目 A 为止。

例子 3

我们可以传递 DOM 元素作为 context，在其中搜索最接近的元素。

```
var listItemII = document.getElementById('ii');
$('li.item-a').closest('ul', listItemII).css('background-color', 'r
$('li.item-a').closest('#one', listItemII).css('background-color',
```

以上代码会改变 level-2 <ul> 的颜色，因为它既是列表项 A 的第一个 <ul> 祖先，同时也是列表项 II 的后代。它不会改变 level-1 <ul> 的颜色，因为它不是 list item II 的后代。

jQuery 遍历 - contents() 方法

实例

找到段落中的所有文本节点，并用粗体标签包装它们。

```
$("p").contents().filter(function(){ return this.nodeType != 1; })
```

定义和用法

contents() 方法获得匹配元素集合中每个元素的子节点，包括文本和注释节点。

语法

```
.contents()
```

详细说明

如果给定表示 DOM 元素集合的 jQuery 对象，.contents() 方法允许我们检索 DOM 树中的这些元素的直接子节点，并用匹配元素构造新的 jQuery 对象。.contents() 和 .children() 方法类似，不同的是前者在结果 jQuery 对象中包含了文本节点以及 HTML 元素。

.contents() 方法也可以用于获得 iframe 的内容文档，前提是该 iframe 与主页面在同一个域。

请思考下面这个带有一些文本节点的 <div>，每个节点被两个折行元素 (
) 分隔：

```
<div class="container">
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
  do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  <br /><br />
  Ut enim ad minim veniam, quis nostrud exercitation ullamco
  laboris nisi ut aliquip ex ea commodo consequat.
  <br /> <br />
  Duis aute irure dolor in reprehenderit in voluptate velit
  esse cillum dolore eu fugiat nulla pariatur.
</div>
```

我们可以使用 .contents() 方法来把文本块转换为形式良好的段落：

```
$('.container').contents().filter(function() {
  return this.nodeType == 3;
})
  .wrap('<p></p>')
.end()
.filter('br')
  .remove();
```

这段代码首先会接收 <div class="container"> 的内容，然后滤过其文本节点，将文本节点封装入段落标签中。这是通过测试元素的 .nodeType 属性实现的。该属性存有指示节点类型的数字代码；文本节点使用代码 3。内容会被再次过滤，这次针对
 元素，这些元素会被移除。

jQuery 遍历 - each() 方法

实例

输出每个 li 元素的文本：

```
$("button").click(function(){
  $("li").each(function(){
    alert($(this).text())
  });
});
```

定义和用法

each() 方法规定为每个匹配元素规定运行的函数。

提示：返回 false 可用于及早停止循环。

语法

```
$(_selector_).each(_function(index,element)_)
```

参数	描述
function(index,element)	必需。为每个匹配元素规定运行的函数。 *index* - 选择器的 index 位置 *element* - 当前的元素（也可使用 "this" 选择器）

jQuery 遍历 - end() 方法

实例

选择所有段落，找到这些段落中的 span 元素，然后将它们恢复为段落，并把段落设置为两像素的红色边框：

```
$("p").find("span").end().css("border", "2px red solid");
```

定义和用法

end() 方法结束当前链条中的最近的筛选操作，并将匹配元素集还原为之前的状态。

语法

```
.end()
```

详细说明

大多数 jQuery 的遍历方法会操作一个 jQuery 对象实例，并生成一个匹配不同 DOM 元素集的新对象。当发生这种情况时，应该会把新的元素集推入维持在对象中的堆栈内。每次成功的筛选方法调用都会把新元素推入堆栈中。如果我们需要老的元素集，可以使用 end() 从堆栈中弹出新集合。

假设页面中有一对很短的列表：

```
<ul class="first">
   <li class="foo">list item 1</li>
   <li>list item 2</li>
   <li class="bar">list item 3</li>
</ul>
<ul class="second">
   <li class="foo">list item 1</li>
   <li>list item 2</li>
   <li class="bar">list item 3</li>
</ul>
```

例子 1

主要是在利用 jQuery 的链条属性（命令链）时，jQuery 会比较有用。如果不使用命令链，我们一般是通过变量名来调用之前的对象，这样我们就不需要操作堆栈了。不过通过 end()，我们可以把所有方法调用串联在一起：

```
$('ul.first').find('.foo').css('background-color', 'red')
  .end().find('.bar').css('background-color', 'green');
```

这条命令链检索第一个列表中类名为 foo 的项目，并把它们的背景设置为红色。end() 会将对象还原为调用 find() 之前的状态，所以第二个 find() 查找的是 <ul class="first"> 内的 '.bar'， 而不是在列表的 <li class="foo"> 中查找，并将匹配元素的背景设置为绿色。最后的结果是第一个列表中的项目 1 和项目 3 被设置了带颜色的背景，而第二个列表中的项目没有任何变化。

例子 2

这条长长的 jQuery 链可以可视化为结构化的代码块，筛选方法打开嵌套代码块，而 end() 方法用来关闭代码块：

```
$('ul.first').find('.foo')
  .css('background-color', 'red')
.end().find('.bar')
  .css('background-color', 'green')
.end();
```

最后这个 end() 不是必需的，因为我们随后会丢弃这个 jQuery 对象。不过，如果按照这种形式编写代码，end() 就能提供视觉上的对称，以及规整程序的感觉，至少对于开发者来说更易阅读，当然代价则是由于进行了额外的调用，会有一点点性能损失。

jQuery 遍历 - eq() 方法

实例

通过为 index 为 2 的 div 添加适当的类，将其变为蓝色：

```
$("body").find("div").eq(2).addClass("blue");
```

定义和用法

eq() 方法将匹配元素集缩减值指定 index 上的一个。

语法

```
.eq(_index_)
```

参数	描述
index	整数，指示元素的位置（最小为 0）。如果是负数，则从集合中的最后一个元素往回计数。

详细说明

如果给定表示 DOM 元素集合的 jQuery 对象，.eq() 方法会用集合中的一个元素构造一个新的 jQuery 对象。所使用的 index 参数标示集合中元素的位置。

请看下面这个简单的列表：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

例子 1

我们可以把该方法应用到这个列表项目集：

```
$('li').eq(2).css('background-color', 'red');
```

这个调用的结果是为项目 3 设置了红色背景。请注意，index 是基于零的，并且是在 jQuery 对象中引用元素的位置，而不是在 DOM 树中。

例子 2

如果提供负数，则指示从集合结尾开始的位置，而不是从开头开始。例如：

```
$('li').eq(-2).css('background-color', 'red');
```

这次，项目 4 的背景变为红色，这是因为它是集合结尾开始的第二个。

例子 3

如果无法根据指定的 index 参数找到元素，则该方法构造带有空集的 jQuery 对象，length 属性为 0。

```
$('li').eq(5).css('background-color', 'red');
```

这里，没有列表项会变为红色，这是因为 .eq(5) 指示的第六个列表项。

jQuery 遍历 - filter() 方法

实例

改变所有 div 的颜色，然后向类名为 "middle" 的类添加边框：

```
$("div").css("background", "#c8ebcc")
  .filter(".middle")
  .css("border-color", "red");
```

定义和用法

filter() 方法将匹配元素集合缩减为匹配指定选择器的元素。

语法

```
.filter(_selector_)
```

参数	描述
selector	字符串值，包含供匹配当前元素集合的选择器表达式。

详细说明

如果给定表示 DOM 元素集合的 jQuery 对象，.filter() 方法会用匹配元素的子集构造一个新的 jQuery 对象。所使用的选择器会测试每个元素；所有匹配该选择器的元素都会包含在结果中。

请思考下面这个拥有简单列表的页面：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
  <li>list item 6</li>
</ul>
```

我们可以向这个列表项集合应用该方法：

```
$('li').filter(':even').css('background-color', 'red');
```

此调用的结果是将项目 1, 3, 5 的背景设置为红色，这是因为它们都匹配选择器（回忆一下，:even 和 :odd 均使用基于 0 的 index）。

使用过滤函数

使用该方法的第二个形式是，通过函数而不是选择器来筛选元素。对于每个元素，如果该函数返回 true，则元素会被包含在已筛选集合中；否则，会排除这个元素。

请看下面这段稍显复杂的 HTML 片段：

```
<ul>
  <li><strong>list</strong> item 1 - one strong tag</li>
  <li><strong>list</strong> item <strong>2</strong>
   - two <span>strong tags</span></li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
  <li>list item 6</li>
</ul>
```

我们可以选取这些列表项，然后基于其内容来筛选它们：

```
$('li').filter(function(index) {
  return $('strong', this).length == 1;
}).css('background-color', 'red');
```

jQuery 遍历 - find() 方法

实例

搜索所有段落中的后代 span 元素，并将其颜色设置为红色：

```
$("p").find("span").css('color','red');
```

定义和用法

find() 方法获得当前元素集合中每个元素的后代，通过选择器、jQuery 对象或元素来筛选。

语法

```
.find(_selector_)
```

参数	描述
selector	字符串值，包含供匹配当前元素集合的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.find() 方法允许我们在 DOM 树中搜索这些元素的后代，并用匹配元素来构造一个新的 jQuery 对象。.find() 与 .children() 方法类似，不同的是后者仅沿着 DOM 树向下遍历单一层级。

.find() 方法第一个明显特征是，其接受的选择器表达式与我们向 $() 函数传递的表达式的类型相同。将通过测试这些元素是否匹配该表达式来对元素进行过滤。

请思考下面这个简单的嵌套列表：

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

我们将从列表 II 开始来查找其中的列表项：

```
$('li.item-ii').find('li').css('background-color', 'red');
```

这次调研的结果是，项目 A、B、1、2、3 以及 C 均被设置了红色背景。即使项目 II 匹配选择器表达式，它也不会被包含在结果中；只会对后代进行匹配。

与其他的树遍历方法不同，选择器表达式对于 .find() 是必需的参数。如果我们需要实现对所有后代元素的取回，可以传递通配选择器 '*'。

选择器 context 是由 .find() 方法实现的；因此，$('li.item-ii').find('li') 等价于 $('li', 'li.item-ii')。

对于 jQuery 1.6，我们还可以使用给定的 jQuery 集合或元素来进行筛选。还是上面的嵌套列表，我们首先这样写：

```
var $allListElements = $('li');
```

然后将这个 jQuery 对象传递给 find 方法：

```
$('li.item-ii').find( $allListElements );
```

上面的代码会返回一个 jQuery 集合，其中包含属于列表 II 后代的列表元素。

类似地，也可以传递一个元素：

```
var item1 = $('li.item-1')[0];
$('li.item-ii').find( item1 ).css('background-color', 'red');
```

这次调用的结果是项目 1 被设置为红色背景。

jQuery 遍历 - first() 方法

实例

高亮显示段落中的第一个 span ：

```
$("p span").first().addClass('highlight');
```

定义和用法

first() 将匹配元素集合缩减为集合中的第一个元素。

语法

```
.first()
```

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.first() 方法会用第一个匹配元素构造一个新的 jQuery 对象。

请思考下面这个带有简单列表的页面：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

我们可以对这个列表项集合应用该方法：

```
$('li').first().css('background-color', 'red');
```

这次调用的结果是，第一个项目被设置为红色背景。

jQuery 遍历 - has() 方法

实例

检测某个元素是否在另一个元素中：

```
$("ul").append("<li>" + ($("ul").has("li").length ? "Yes" : "No") -
$("ul").has("li").addClass("full");
```

定义和用法

has() 将匹配元素集合缩减为拥有匹配指定选择器或 DOM 元素的后代的子集。

语法

```
.has(_selector_)
```

参数	描述
selector	字符串值，包含匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.has() 方法用匹配元素的子集来构造一个新的 jQuery 对象。所使用的选择器用于检测匹配元素的后代；如果任何后代元素匹配该选择器，该元素将被包含在结果中。

请思考下面这个带有嵌套列表的页面：

```
<ul>
  <li>list item 1</li>
  <li>list item 2
    <ul>
      <li>list item 2-a</li>
      <li>list item 2-b</li>
    </ul>
  </li>
  <li>list item 3</li>
  <li>list item 4</li>
</ul>
```

我们可以对列表项集合应用该方法，就像这样：

```
$('li').has('ul').css('background-color', 'red');
```

该调用的结果是，项目 2 的背景被设置为红色，这是因为该项目是后代中唯一拥有 <ul> 的 。

jQuery 遍历 - is() 方法

实例

返回 false，因为 input 元素的父元素是 p 元素：

```
var isFormParent = $("input[type='checkbox']").parent().is("form'
$("div").text("isFormParent = " + isFormParent);
```

定义和用法

is() 根据选择器、元素或 jQuery 对象来检测匹配元素集合，如果这些元素中至少有一个元素匹配给定的参数，则返回 true。

语法

```
.is(_selector_)
```

参数	描述
selector	字符串值，包含匹配元素的选择器表达式。

详细说明

与其他筛选方法不同，.is() 不创建新的 jQuery 对象。相反，它允许我们在不修改 jQuery 对象内容的情况下对其进行检测。这在 callback 内部通常比较有用，比如事件处理程序。

假设我们有一个列表，其中两个项目包含子元素：

```
<ul>
  <li>list <strong>item 1</strong></li>
  <li><span>list item 2</span></li>
  <li>list item 3</li>
</ul>
```

您可以向 <ul> 元素添加 click 处理程序，然后把代码限制为只有当列表项本身，而非子元素，被点击时才进行触发：

```
$("ul").click(function(event) {
  var $target = $(event.target);
  if ( $target.is("li") ) {
    $target.css("background-color", "red");
  }
});
```

现在，当用户点击的是第一个列表项中的单词 "list" 或第三个列表项中的任何单词时，被点击的列表项会被设置为红色背景。不过，当用户点击第一个列表项中的 item 1 或第二个列表项中的任何单词时，都不会有任何变化，这是因为这上面的情况中，事件的目标分别是 <strong> 是 <span>。

请您注意，对于带有位置性选择器的选择器表达式字符串，比如 :first、:gt() 或者 :even，位置性筛选是针对传递到 .is() 的 jQuery 对象进行的，而非针对包含文档。所以对于上面的 HTML 来说，诸如 $("li:first").is("li:last") 的表达式返回 true，但是 $("li:first-child").is("li:last-child") 返回 false。

使用函数

该方法的第二种用法是，对基于函数而非选择器的相关元素的表达式进行求值。对于每个元素来说，如果该函数返回 true，则 .is() 也返回 true。例如，下面是稍微复杂的 HTML 片段：

```
<ul>
  <li><strong>list</strong> item 1 - one strong tag</li>
  <li><strong>list</strong> item <strong>2</strong> -
    two <span>strong tags</span></li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

您可以向每个 <li> 添加 click 处理程序，以计算在被点击的 <li> 内部 <strong> 元素的数目：

```
$("li").click(function() {
  var $li = $(this),
    isWithTwo = $li.is(function() {
      return $('strong', this).length === 2;
    });
  if ( isWithTwo ) {
    $li.css("background-color", "green");
  } else {
    $li.css("background-color", "red");
  }
});
```

jQuery 遍历 - last() 方法

实例

高亮显示段落中的最后一个 span ：

```
$("p span").last().addClass('highlight');
```

定义和用法

last() 将匹配元素集合缩减为集合中的最后一个元素。

语法

```
.last()
```

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.last() 方法会用最后一个匹配元素构造一个新的 jQuery 对象。

请思考下面这个带有简单列表的页面：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

我们可以对这个列表项集合应用该方法：

```
$('li').last().css('background-color', 'red');
```

这次调用的结果是，最后一个项目被设置为红色背景。

jQuery 遍历 - map() 方法

实例

构建表单中所有值的列表：

```
$("p").append( $("input").map(function(){
  return $(this).val();
}).get().join(", ") );
```

定义和用法

map() 把每个元素通过函数传递到当前匹配集合中，生成包含返回值的新的 jQuery 对象。

语法

```
.map(_callback(index,domElement)_)
```

参数	描述
callback(index,domElement)	对当前集合中的每个元素调用的函数对象。

详细说明

由于返回值是 jQuery 封装的数组，使用 get() 来处理返回的对象以得到基础的数组。

.map() 方法对于获得或设置元素集的值特别有用。请思考下面这个带有一系列复选框的表单：

```
<form method="post" action="">
  <fieldset>
    <div>
      <label for="two">2</label>
      <input type="checkbox" value="2" id="two" name="number[]">
    </div>
    <div>
      <label for="four">4</label>
      <input type="checkbox" value="4" id="four" name="number[]">
    </div>
    <div>
      <label for="six">6</label>
      <input type="checkbox" value="6" id="six" name="number[]">
    </div>
    <div>
      <label for="eight">8</label>
      <input type="checkbox" value="8" id="eight" name="number[]">
    </div>
  </fieldset>
</form>
```

我们能够获得复选框 ID 组成的逗号分隔的列表：

```
$(':checkbox').map(function() {
  return this.id;
}).get().join(',');
```

本次调用的结果是字符串："two,four,six,eight"。

在 callback 函数内部，this 引用每次迭代的当前 DOM 元素。该函数可返回单独的数据项，或者是要被插入结果集中的数据项的数组。如果返回的是数组，数组内的元素会被插入集合中。如果函数返回 null 或 undefined，则不会插入任何元素。

jQuery 遍历 - next() 方法

实例

查找每个段落的下一个同胞元素，仅选中类名为 "selected" 的段落：

```
$("p").next(".selected").css("background", "yellow");
```

定义和用法

next() 获得匹配元素集合中每个元素紧邻的同胞元素。如果提供选择器，则取回匹配该选择器的下一个同胞元素。

语法

```
.next(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.next() 方法允许我们搜索 DOM 树中的元素紧跟的同胞元素，并用匹配元素构造新的 jQuery 对象。

该方法接受可选的选择器表达式，类型与我传递到 $() 函数中的相同。如果紧跟的同胞元素匹配该选择器，则会留在新构造的 jQuery 对象中；否则会将之排除。

请思考下面这个带有简单列表的页面：

```
<ul>
   <li>list item 1</li>
   <li>list item 2</li>
   <li class="third-item">list item 3</li>
   <li>list item 4</li>
   <li>list item 5</li>
</ul>
```

如果我们从项目三开始，则能够找到其后出现的元素：

```
$('li.third-item').next().css('background-color', 'red');
```

这次调用的结果是，项目 4 被设置为红色背景。由于我们没有应用选择器表达式，紧跟的这个元素很明确地被包括为对象的一部分。如果我们已经应用了选择器，在包含它之前会检测是否匹配。

jQuery 遍历 - nextAll() 方法

实例

查找第一个 div 之后的所有类名，并为他们添加类名：

```
$("div:first").nextAll().addClass("after");
```

定义和用法

nextAll() 获得匹配元素集合中每个元素的所有跟随的同胞元素，由选择器筛选是可选的。

语法

```
.nextAll(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.nextAll() 方法允许我们搜索 DOM 树中的元素跟随的同胞元素，并用匹配元素构造新的 jQuery 对象。

该方法接受可选的选择器表达式，类型与我传递到 $() 函数中的相同。如果应用选择器，则将通过检测元素是否匹配来对它们进行筛选。

请思考下面这个带有简单列表的页面：

```
<ul>
   <li>list item 1</li>
   <li>list item 2</li>
   <li class="third-item">list item 3</li>
   <li>list item 4</li>
   <li>list item 5</li>
</ul>
```

如果我们从项目三开始，那么我们能够找到其后出现的元素：

```
$('li.third-item').nextAll().css('background-color', 'red');
```

这次调用的结果是，项目 4 和 5 被设置为红色背景。由于我们没有应用选择器表达式，紧跟的这个元素很明确地被包括为对象的一部分。如果我们已经应用了选择器，在包含它之前会检测是否匹配。

jQuery 遍历 - nextUntil() 方法

实例

查找跟随 <dt id="term-2"> 的同胞元素，直到下一个 <dt>，然后将它们设置为红色背景色。同时，找到跟随 <dt id="term-1"> 的 <dd> 同胞元素，直到 <dt id="term-3">，并为它们设置蓝色文本颜色。

```
$("#term-2").nextUntil("dt").css("background-color", "red");
var term3 = document.getElementById("term-3");
$("#term-1").nextUntil(term3, "dd").css("color", "blue");
```

定义和用法

nextUntil() 获得每个元素所有跟随的同胞元素，但不包括被选择器、DOM 节点或已传递的 jQuery 对象匹配的元素。

语法 1

```
.nextUntil(_selector_,_filter_)
```

参数	描述
selector	字符串值，包含指示在何处停止匹配跟随的同胞元素的选择器表达式。
filter	字符串值，包含用于匹配元素的选择器表达式。

语法 2

```
.nextUntil(_element_,_filter_)
```

参数	描述
element	指示在何处停止匹配跟随的同胞元素的 DOM 节点或 jQuery 对象。
filter	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.nextUntil() 方法允许我们搜索 DOM 树中的元素跟随的同胞元素，当遇到被该方法的参数所匹配的元素时会停止搜索。返回的新 jQuery 对象包含所有跟随的同胞元素，但不包含被参数匹配的元素。

如果选择器不匹配或未规定选择器，则会选取所有跟随的同胞；如果不提供供筛选的选择器，则该方法选取的元素与 .nextAll() 方法相同。

对于 jQuery 1.6，DOM 节点或 jQuery 对象，而不是选择器，可传递到 .nextUntil() 方法。

该方法接受可选的选择器表达式作为其第二参数。如果指定该参数，则将通过检测元素是否匹配该选择器来筛选它们。

jQuery 遍历 - not() 方法

实例

从包含所有段落的集合中删除 id 为 "selected" 的段落：

```
$("p").not("#selected")
```

定义和用法

not() 从匹配元素集合中删除元素。

语法 1

```
.not(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

语法 2

```
.not(_element_)
```

参数	描述
element	一个或多个需要从匹配集中删除的 DOM 元素。

语法 3

```
.not(_function(index)_)
```

参数	描述
function(index)	用于检测集合中每个元素的函数。this 是当前 DOM 元素。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.not() 方法会用匹配元素的子集构造一个新的 jQuery 对象。所应用的选择器会检测每个元素；不匹配该选择器的元素会被包含在结果中。

请思考下面这个带有简单列表的页面：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

我们可以向列表项集应用该方法：

```
$('li').not(':even').css('background-color', 'red');
```

这次调用的结果是将项目 2 和 4 设置为红色背景，这是因为它们不匹配选择器（回忆一下，:even 和 :odd 均使用基于 0 的 index）。

移除具体的元素

.not() 方法的第二个版本允许我们从匹配集中删除元素，假设我们之前已经通过其他手段找到了这些元素。例如，设想一个列表已经将 id 应用到其中一个项目中：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li id="notli">list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

我们可以使用原生的 JavaScript 函数 getElementById() 读取第三个列表项，然后把它从 jQuery 对象中删除：

```
$('li').not(document.getElementById('notli')).css('background-color
```

这条语句改变项目 1、2、3 和 5 的背景色。我们可以用更简单的 jQuery 表达式来完成同样的事情，但是这项技术在比方说其他库提供对纯 DOM 节点的引用时会很有用。

对于 jQuery 1.4，.not() 方法能够采用函数作为其参数，与 .filter() 方法相同。其函数返回 true 的元素会被排除在过滤集之外；所有其他元素将被包含其中。

jQuery 遍历 - offsetParent() 方法

实例

设置类名为 item-a 的 li 元素的最近定位父元素的背景色：

```
$('li.item-a').offsetParent().css('background-color', 'red');
```

定义和用法

offsetParent() 获得被定位的最近祖先元素。

语法

```
.offsetParent()
```

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.offsetParent() 方法允许我们搜索 DOM 树中元素的祖先，并构造一个由最近的定位祖先元素包围的 jQuery 对象。定位元素指的是，元素的 CSS position 属性设置为 relative、absolute 或 fixed。在为表演动画计算偏移或在页面上放置对象时，该信息会很有用处。

请思考带有基本嵌套列表的页面，其中带有定位元素：

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii" style="position: relative;">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

如果我们从项目 A 开始，我们可以找到其定位祖先元素：

```
$('li.item-a').offsetParent().css('background-color', 'red');
```

这会改变被定位的项目 II 的背景色。

jQuery 遍历 - parent() 方法

实例

查找每个段落的带有 "selected" 类的父元素：

```
$("p").parent(".selected")
```

定义和用法

parent() 获得当前匹配元素集合中每个元素的父元素，使用选择器进行筛选是可选的。

```
.parent(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.parent() 方法允许我们在 DOM 树中搜索这些元素的父元素，并用匹配元素构造一个新的 jQuery 对象。.parents() 和 .parent() 方法类似，不同的是后者沿 DOM 树向上遍历单一层级。

该方法接受可选的选择器表达式，与我们向 $() 函数中传递的参数类型相同。如果应用这个选择器，则将通过检测元素是否匹配该选择器对元素进行筛选。

请思考这个带有基本的嵌套列表的页面：

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

如果我们从项目 A 开始，则可找到其父元素：

```
$('li.item-a').parent().css('background-color', 'red');
```

此次调用的结果是，为 level-2 列表设置红色背景。由于我们未应用选择器表达式，父元素很自然地成为了对象的一部分。如果已应用选择器，则会在包含元素之前，检测元素是否匹配选择器。

jQuery 遍历 - parents() 方法

实例

查找每个 b 元素的所有父元素：

```
$("b").parents()
```

定义和用法

parents() 获得当前匹配元素集合中每个元素的祖先元素，使用选择器进行筛选是可选的。

```
.parents(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.parents() 方法允许我们在 DOM 树中搜索这些元素的祖先元素，并用从最近的父元素向上的顺序排列的匹配元素构造一个新的 jQuery 对象。元素是按照从最近的父元素向外的顺序被返回的。.parents() 和 .parent() 方法类似，不同的是后者沿 DOM 树向上遍历单一层级。

该方法接受可选的选择器表达式，与我们向 $() 函数中传递的参数类型相同。如果应用这个选择器，则将通过检测元素是否匹配该选择器对元素进行筛选。

请思考这个带有基本的嵌套列表的页面：

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

如果我们从项目 A 开始，则可找到其祖先元素：

```
$('li.item-a').parents().css('background-color', 'red');
```

此次调用的结果是，level-2 列表、项目 II 以及 level-1 列表等元素（沿 DOM 树一路向上直到 <html>）设置红色背景。由于我们未应用选择器表达式，父元素很自然地成为了对象的一部分。如果已应用选择器，则会在包含元素之前，检测元素是否匹配选择器。由于我们未应用选择器表达式，所有祖先元素都是返回的 jQuery 对象的组成部分。如果已应用选择器，则只会包含其中的匹配项目。

jQuery 遍历 - parentsUntil() 方法

实例

查找 <li class="item-a"> 的祖先元素，直到 <ul class="level-1">，并将它们设置为红色背景。同时，找到 <li class="item-2"> 的所有类名为 "yes" 的祖先元素，直到 <ul class="level-1">，然后为它们设置蓝色边框：

```
$("li.item-a").parentsUntil(".level-1")
  .css("background-color", "red");

$("li.item-2").parentsUntil( $("ul.level-1"), ".yes" )
  .css("border", "3px solid blue");
```

定义和用法

parentsUntil() 获得当前匹配元素集合中每个元素的祖先元素，直到（但不包括）被选择器、DOM 节点或 jQuery 对象匹配的元素。

语法 1

```
.parentsUntil(_selector_,_filter_)
```

参数	描述
selector	可选。字符串值，规定在何处停止对祖先元素进行匹配的选择器表达式。
filter	可选。字符串值，包含用于匹配元素的选择器表达式。

语法 2

```
.parentsUntil(_element_,_filter_)
```

参数	描述
element	可选。DOM 节点或 jQuery 对象，指示在何处停止对祖先元素的匹配。
filter	可选。字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.parentsUntil() 方法允许我们在 DOM 树中搜索这些元素的祖先元素，直到遇到被选择器（传递到方法中的参数）匹配的元素为止。返回的 jQuery 对象包含所有祖先元素，但不包括由 .parentsUntil() 方法规定的选择器匹配的那个元素。

如果不匹配或未应用选择器，则将选区所有祖先元素；在这种情况下，该方法选取的元素与未提供选择器时的 .parents() 相同。

对于 jQuery 1.6，DOM 节点或 jQuery 对象，而不是选择器，可用作 .parentsUntil() 方法的第一个参数。

该方法接受可选的选择器表达式作为其第二参数。如果应用这个参数，则将通过检测元素是否匹配该选择器对元素进行筛选。

jQuery 遍历 - prev() 方法

实例

检索每个段落，找到类名为 "selected" 的前一个同胞元素：

```
$("p").prev(".selected")
```

定义和用法

prev() 获得匹配元素集合中每个元素紧邻的前一个同胞元素，通过选择器进行筛选是可选的。

```
.prev(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.prev() 方法允许我们在 DOM 树中搜索这些元素的前一个同胞元素，并用匹配元素构造一个新的 jQuery 对象。

该方法接受可选的选择器表达式，与我们向 $() 函数中传递的参数类型相同。如果应用这个选择器，则将通过检测元素是否匹配该选择器对元素进行筛选。

请思考这个带有基本的嵌套列表的页面：

```
<ul>
   <li>list item 1</li>
   <li>list item 2</li>
   <li class="third-item">list item 3</li>
   <li>list item 4</li>
   <li>list item 5</li>
</ul>
```

如果我们从第三个项目开始，则可找到该元素之间的紧邻元素：

```
$('li.third-item').prev().css('background-color', 'red');
```

此处调用的结果是将项目 2 设置为红色背景。由于我们未应用选择器表达式，前一个元素很自然地成为了对象的一部分。如果已应用选择器，则会在包含元素之前，检测元素是否匹配选择器。

jQuery 遍历 - prevAll() 方法

实例

定位最后一个 div 之前的所有 div，并为它们添加类：

```
$("div:last").prevAll().addClass("before");
```

定义和用法

prevAll() 获得当前匹配元素集合中每个元素的前面的同胞元素，使用选择器进行筛选是可选的。

语法

```
.prevAll(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.prevAll() 方法允许我们在 DOM 树中搜索这些元素前面的同胞元素，并用匹配元素构造一个新的 jQuery 对象。

该方法接受可选的选择器表达式，与我们向 $() 函数中传递的参数类型相同。如果应用这个选择器，则将通过检测元素是否匹配该选择器对元素进行筛选。

请思考这个带有基本的嵌套列表的页面：

```
<ul>
   <li>list item 1</li>
   <li>list item 2</li>
   <li class="third-item">list item 3</li>
   <li>list item 4</li>
   <li>list item 5</li>
</ul>
```

如果我们从第三个项目开始，则可找到该元素之间的同胞元素：

```
$('li.third-item').prevAll().css('background-color', 'red');
```

此处调用的结果是将项目 2 和项目 1 设置为红色背景。由于我们未应用选择器表达式，这些前面的元素很自然地成为了对象的一部分。如果已应用选择器，则会在包含元素之前，检测这些元素是否匹配选择器。

jQuery 遍历 - prevUntil() 方法

实例

查找 <dt id="term-2"> 之前的同胞元素，直到前一个 <dt>，并将它们设置为红色。同时，查找 <dt id="term-3"> 前面的 <dd> 同胞，直到 <dt id="term-1">，并把它们设置为蓝色文本：

```
$("#term-2").prevUntil("dt").css("background-color", "red");
var term1 = document.getElementById('term-1');
$("#term-3").prevUntil(term1, "dd").css("color", "green");
```

定义和用法

prevUntil() 方法获得当前匹配元素集合中每个元素的前面的同胞元素，但不包括被选择器、DOM 节点或 jQuery 对象匹配的元素。

语法 1

```
.prevUntil(_selector_, _filter_)
```

参数	描述
selector	可选。字符串值，包含指示在何处停止匹配前方同胞元素的选择器表达式。
filter	可选。字符串值，包含用于匹配元素的选择器表达式。

语法 2

```
.prevUntil(_element_, _filter_)
```

参数	描述
element	可选。指示在何处停止匹配前方同胞元素的 DOM 节点或 jQuery 对象。
filter	可选。字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.prevUntil() 方法允许我们在 DOM 树中搜索这些元素前面的同胞元素，直到遇到被选择器（传递到方法中的参数）匹配的元素为止。返回的 jQuery 新对象包含所有前面的同胞元素，但不包括由 .prevUntil() 方法规定的选择器匹配的那个元素；所返回元素的顺序是从最近的同胞元素到最远的那个。

如果不匹配或未应用选择器，则将选区所有前面的同胞元素；在这种情况下，该方法选取的元素与未提供选择器时的 .prevAll() 相同。

对于 jQuery 1.6，DOM 节点或 jQuery 对象，而不是选择器，可用作 .prevUntil() 方法的第一个参数。

该方法接受可选的选择器表达式作为其第二参数。如果应用这个参数，则将通过检测元素是否匹配该选择器对元素进行筛选。

jQuery 遍历 - siblings() 方法

实例

查找每个 p 元素的所有类名为 "selected" 的所有同胞元素：

```
$("p").siblings(".selected")
```

定义和用法

siblings() 获得匹配集合中每个元素的同胞，通过选择器进行筛选是可选的。

语法

```
.siblings(_selector_)
```

参数	描述
selector	字符串值，包含用于匹配元素的选择器表达式。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.siblings() 方法允许我们在 DOM 树中搜索这些元素的同胞元素，并用匹配元素构造一个新的 jQuery 对象。

该方法接受可选的选择器表达式，与我们向 $() 函数中传递的参数类型相同。如果应用这个选择器，则将通过检测元素是否匹配该选择器对元素进行筛选。

请思考这个带有基本的嵌套列表的页面：

```
<ul>
   <li>list item 1</li>
   <li>list item 2</li>
   <li class="third-item">list item 3</li>
   <li>list item 4</li>
   <li>list item 5</li>
</ul>
```

如果我们从第三个项目开始，则可找到该元素的同胞元素：

```
$('li.third-item').siblings().css('background-color', 'red');
```

此处调用的结果是将项目 1、2、4 和 5 的背景设置为红色。 设置为红色背景。由于我们未应用选择器表达式，所有同胞元素很自然地成为了对象的一部分。如果已应用选择器，则只会包含四个列表中的匹配的项目。

原始元素不包含在同胞元素中，当我们打算找到 DOM 树的特定层级上的所有元素时，记住一点很重要。

jQuery 遍历 - slice() 方法

实例

选中所有段落，然后将所选内容缩减为只包含第一和第二个段落：

```
$("p").slice(0, 2).wrapInner("");
```

定义和用法

slice() 把匹配元素集合缩减为指定的指数范围的子集。

语法

```
.slice(_selector_,_end_)
```

参数	描述
selector	基于 0 的整数值，指示开始选取元素的位置。如果是负数，则指示从集合末端开始的偏移量。
end	基于 0 的整数值，指示结束选取元素的位置。如果是负数，则指示从集合末端开始的偏移量。如果省略，则选取范围会在集合末端结束。

详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.slice() 方法用匹配元素的子集构造一个新的 jQuery 对象。已应用的 index 参数集合中其中一个元素的位置；如果省略 end 参数，则 index 之后的所有的所有元素都会包含在结果中。

请思考这个带有简单列表的页面：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

我们可以向列表项集合应用该方法：

```
$('li').slice(2).css('background-color', 'red');
```

此次调用的结果是项目 3、4 和 5 的背景被设置为红色。请注意，已应用的 index 参数基于零，引用的是 jQuery 对象中元素的位置，而非 DOM 树中的。

end 参数允许我们更进一步地限制选取范围。比如：

```
$('li').slice(2, 4).css('background-color', 'red');
```

现在，只有项目 3 和 4 会被选取。再次说明，index 是基于零的；范围会延伸到（但不包含）指定的 index。

负的指数

jQuery 的 .slice() 方法模仿了 JavaScript 数组对象的 .slice() 方法。它所模仿的特性之一是向 start 或 end 参数传递负数的能力。如果提供负数，则指示的是从集合结尾开始的一个位置，而非从开头。例如：

```
$('li').slice(-2, -1).css('background-color', 'red');
```

这次，只有列表项 4 会变红，这是因为该项目是介于从结尾计数的二 (-2) 与从结尾计数的一 (-1) 的之间的范围中的唯一项目。

jQuery 参考手册 - 数据

jQuery 数据操作函数

这些方法允许我们将指定的 DOM 元素与任意数据相关联。

函数	描述
.clearQueue()	从队列中删除所有未运行的项目。
.data()	存储与匹配元素相关的任意数据。
jQuery.data()	存储与指定元素相关的任意数据。
.dequeue()	从队列最前端移除一个队列函数，并执行它。
jQuery.dequeue()	从队列最前端移除一个队列函数，并执行它。
jQuery.hasData()	存储与匹配元素相关的任意数据。
.queue()	显示或操作匹配元素所执行函数的队列。
jQuery.queue()	显示或操作匹配元素所执行函数的队列。
.removeData()	移除之前存放的数据。
jQuery.removeData()	移除之前存放的数据。

参阅

参考手册：jQuery 队列控制

jQuery 遍历 - clearQueue() 方法

实例

清空队列：

```
$("div").clearQueue();
```

定义和用法

clearQueue() 方法从序列中删除仍未运行的所有项目。

语法

```
.clearQueue(_queueName_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

详细说明

当调用 .clearQueue() 方法时，序列中未被执行的所有函数都会被从序列中删除。如果不使用参数，则 .clearQueue() 从 fx （标准效果序列）中删除剩余的函数。在这种方式中，它类似于 .stop(true)。不过，.stop() 方法只用于动画，而 .clearQueue() 也可用于删除通过 .queue() 方法添加到通用 jQuery 序列的任何函数。

jQuery 数据 - data() 方法

实例

向元素附加数据，然后取回该数据：

```
$("#btn1").click(function(){
  $("div").data("greeting", "Hello World");
});
$("#btn2").click(function(){
  alert($("div").data("greeting"));
});
```

定义和用法

data() 方法向被选元素附加数据，或者从被选元素获取数据。

从元素返回数据

从被选元素中返回附加的数据。

语法

```
$(_selector_).data(_name_)
```

参数	描述
name	可选。规定要取回的数据的名称。如果没有规定名称，则该方法将以对象的形式从元素中返回所有存储的数据。

向元素附加数据

向被选元素附加数据。

语法

```
$(_selector_).data(_name_,_value_)
```

参数	描述
name	必需。规定要设置的数据的名称。
value	必需。规定要设置的数据的值。

使用对象向元素附加数据

使用带有名称/值对的对象向被选元素添加数据。

语法

```
$(_selector_).data(_object_)
```

参数	描述
object	必需。规定包含名称/值对的对象。

jQuery 数据 - jQuery.data() 方法

实例

向元素附加数据，然后取回该数据：

```
$("#btn1").click(function(){
  $("div").data("greeting", "Hello World");
});
$("#btn2").click(function(){
  alert($("div").data("greeting"));
});
```

定义和用法

data() 方法向被选元素附加数据，或者从被选元素获取数据。

注释：这是底层级的方法；使用 .data() 更加方便。

从元素返回数据

从被选元素中返回附加的数据。

语法

```
$(_selector_).data(_name_)
```

参数	描述
name	可选。规定要取回的数据的名称。如果没有规定名称，则该方法将以对象的形式从元素中返回所有存储的数据。

向元素附加数据

向被选元素附加数据。

语法

```
$(_selector_).data(_name_,_value_)
```

参数	描述
name	必需。规定要设置的数据的名称。
value	必需。规定要设置的数据的值。

使用对象向元素附加数据

使用带有名称/值对的对象向被选元素添加数据。

语法

```
$(_selector_).data(_object_)
```

参数	描述
object	必需。规定包含名称/值对的对象。

jQuery 遍历 - dequeue() 方法

实例

使用 dequeue() 终止一个自定义的队列函数：

```
$("div").queue(function () {
  $(this).toggleClass("red");
  $(this).dequeue();
});
```

定义和用法

dequeue() 方法为匹配元素执行序列中的下一个函数。

语法

```
.dequeue(_queueName_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

详细说明

当调用 .dequeue() 时，会从序列中删除下一个函数，然后执行它。该函数反过来会（直接或间接地）引发对 .dequeue() 的调用，这样序列才能继续下去。

jQuery 遍历 - jQuery.dequeue() 方法

实例

使用 dequeue() 终止一个自定义的队列函数：

```
$("div").queue(function () {
  $(this).toggleClass("red");
  $(this).dequeue();
});
```

定义和用法

dequeue() 方法为匹配元素执行序列中的下一个函数。

注释：这是底层级的方法；使用 .dequeue() 更加方便。

语法

```
.dequeue(_queueName_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

详细说明

当调用 .dequeue() 时，会从序列中删除下一个函数，然后执行它。该函数反过来会（直接或间接地）引发对 .dequeue() 的调用，这样序列才能继续下去。

jQuery 遍历 - hasData() 方法

实例

在元素上设置数据，然后查看 hasData 的结果：

```
$(function(){
  var $p = jQuery("p"), p = $p[0];
  $p.append(jQuery.hasData(p)+" "); /* false */
  jQuery.data(p, "testing", 123);
  $p.append(jQuery.hasData(p)+" "); /* true */
  jQuery.removeData(p, "testing");
  $p.append(jQuery.hasData(p)+" "); /* false */
});
```

定义和用法

hasData() 方法检测元素是否拥有与之相关的任何 jQuery 数据。

语法

```
jQuery.hasData(_element_)
```

参数	描述
element	可选。需要检查其数据的 DOM 元素。

详细说明

jQuery.hasData() 方法检测元素当前是否拥有通过使用 jQuery.data() 设置的任何值。如果没有数据与元素相关（根本不存在数据对象或者数据对象为空），则该方法返回 false；否则返回 true。

jQuery.hasData(*element*) 的主要优势是，在不存在数据对象的情况下，不会创建并将数据对象与元素进行关联。相反地，jQuery.data(*element*) 总是向调用者返回数据对象，如果之前数据对象不存在，则会创建它。

jQuery 遍历 - queue() 方法

实例

显示队列的长度：

```
function showIt() {
  var n = div.queue("fx");
  $("span").text( n.length );
  setTimeout(showIt, 100);
}
```

定义和用法

queue() 方法显示或操作在匹配元素上执行的函数队列。

语法

```
.queue(_queueName_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

操作队列

queue() 方法操作在匹配元素上执行的函数队列。

语法

```
.queue(_queueName_,_newQueue_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

详细说明

每个元素均可拥有一到多个由 jQuery 添加的函数队列。在大多数应用程序中，只使用一个队列（名为 fx）。队列运行在元素上异步地调用动作序列，而不会终止程序执行。典型例子时调用元素上的多个动画方法。例如：

```
$('#foo').slideUp().fadeIn();
```

当这条语句执行时，元素会立即开始其滑动动画，但是淡入过渡被置于 fx 队列，只有当滑动过渡完成后才会被调用。

.queue() 方法允许我们直接对这个函数队列进行操作。调用带有回调函数的 .queue() 方法特别有用；它允许我们在队列末端放置一个新函数。

这个特性与动画方法提供回调函数类似，但是无需在动画执行时设置回调函数。

```
$('#foo').slideUp();
$('#foo').queue(function() {
  alert('Animation complete.');
  $(this).dequeue();
});
```

等价于：

```
$('#foo').slideUp(function() {
  alert('Animation complete.');
});
```

请注意，当通过 .queue() 添加函数时，我们应当确保最终调用了 .dequeue()，这样下一个排队的函数才能执行。

例子 1

对自定义函数进行队列操作：

```
$(document.body).click(function () {
  $("div").show("slow");
  $("div").animate({left:'+=200'},2000);
  $("div").queue(function () {
    $(this).addClass("newcolor");
    $(this).dequeue();
  });
  $("div").animate({left:'-=200'},500);
  $("div").queue(function () {
    $(this).removeClass("newcolor");
    $(this).dequeue();
  });
  $("div").slideUp();
});
```

例子 2

设置队列数组来删除队列：

```
$("#start").click(function () {
  $("div").show("slow");
  $("div").animate({left:'+=200'},5000);
  $("div").queue(function () {
    $(this).addClass("newcolor");
    $(this).dequeue();
  });
  $("div").animate({left:'-=200'},1500);
  $("div").queue(function () {
    $(this).removeClass("newcolor");
    $(this).dequeue();
  });
  $("div").slideUp();
});

$("#stop").click(function () {
  $("div").queue("fx", []);
  $("div").stop();
});
```

jQuery 遍历 - jQuery.queue() 方法

实例

显示队列的长度：

```
function showIt() {
  var n = div.queue("fx");
  $("span").text( n.length );
  setTimeout(showIt, 100);
}
```

定义和用法

queue() 方法显示或操作在匹配元素上执行的函数队列。

注释：这是底层级的方法；使用 .queue() 更加方便。

语法

```
.queue(_queueName_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

操作队列

queue() 方法操作在匹配元素上执行的函数队列。

语法

```
.queue(_queueName_,_newQueue_)
```

参数	描述
queueName	可选。字符串值，包含序列的名称。默认是 fx, 标准的效果序列。

详细说明

每个元素均可拥有一到多个由 jQuery 添加的函数队列。在大多数应用程序中，只使用一个队列（名为 fx）。队列运行在元素上异步地调用动作序列，而不会终止程序执行。典型例子时调用元素上的多个动画方法。例如：

```
$('#foo').slideUp().fadeIn();
```

当这条语句执行时，元素会立即开始其滑动动画，但是淡入过渡被置于 fx 队列，只有当滑动过渡完成后才会被调用。

.queue() 方法允许我们直接对这个函数队列进行操作。调用带有回调函数的 .queue() 方法特别有用；它允许我们在队列末端放置一个新函数。

这个特性与动画方法提供回调函数类似，但是无需在动画执行时设置回调函数。

```
$('#foo').slideUp();
$('#foo').queue(function() {
  alert('Animation complete.');
  $(this).dequeue();
});
```

等价于：

```
$('#foo').slideUp(function() {
  alert('Animation complete.');
});
```

请注意，当通过 .queue() 添加函数时，我们应当确保最终调用了 .dequeue()，这样下一个排队的函数才能执行。

例子 1

对自定义函数进行队列操作：

```
$(document.body).click(function () {
  $("div").show("slow");
  $("div").animate({left:'+=200'},2000);
  $("div").queue(function () {
    $(this).addClass("newcolor");
    $(this).dequeue();
  });
  $("div").animate({left:'-=200'},500);
  $("div").queue(function () {
    $(this).removeClass("newcolor");
    $(this).dequeue();
  });
  $("div").slideUp();
});
```

例子 2

设置队列数组来删除队列：

```
$("#start").click(function () {
  $("div").show("slow");
  $("div").animate({left:'+=200'},5000);
  $("div").queue(function () {
    $(this).addClass("newcolor");
    $(this).dequeue();
  });
  $("div").animate({left:'-=200'},1500);
  $("div").queue(function () {
    $(this).removeClass("newcolor");
    $(this).dequeue();
  });
  $("div").slideUp();
});

$("#stop").click(function () {
  $("div").queue("fx", []);
  $("div").stop();
});
```

jQuery 数据 - removeData() 方法

实例

从元素中删除之前添加的数据：

```
$("#btn2").click(function(){
  $("div").removeData("greeting");
  alert("Greeting is: " + $("div").data("greeting"));
});
```

定义和用法

removeData() 方法删除之前通过 data() 方法设置的数据。

语法

```
$(_selector_).removeData(_name_)
```

参数	描述
name	可选。规定要删除的数据的名称。如果没有规定名称，该方法将从被选元素中删除所有已存储的数据。

jQuery 数据 - jQuery.removeData() 方法

实例

从元素中删除之前添加的数据：

```
$("#btn2").click(function(){
  $("div").removeData("greeting");
  alert("Greeting is: " + $("div").data("greeting"));
});
```

定义和用法

removeData() 方法删除之前通过 data() 方法设置的数据。

注释：这是底层级的方法；使用 .removeData() 更加方便。

语法

```
$(_selector_).removeData(_name_)
```

参数	描述
name	可选。规定要删除的数据的名称。如果没有规定名称，该方法将从被选元素中删除所有已存储的数据。

jQuery 参考手册 - DOM 元素方法

jQuery DOM 元素方法

函数	描述
.get()	获得由选择器指定的 DOM 元素。
.index()	返回指定元素相对于其他指定元素的 index 位置。
.size()	返回被 jQuery 选择器匹配的元素的数量。
.toArray()	以数组的形式返回 jQuery 选择器匹配的元素。

jQuery DOM 元素方法 - get() 方法

实例

获得第一个 p 元素的名称和值：

```
$("button").click(function(){
  x=$("p").get(0);
  $("div").text(x.nodeName + ": " + x.innerHTML);
});
```

定义和用法

get() 方法获得由选择器指定的 DOM 元素。

语法

```
$(_selector_).get(_index_)
```

参数	描述
index	可选。规定获取哪个匹配元素（通过 index 编号）。

jQuery DOM 元素方法 - index() 方法

实例

获得第一个 p 元素的名称和值：

```
$("li").click(function(){
  alert($(this).index());
});
```

定义和用法

index() 方法返回指定元素相对于其他指定元素的 index 位置。

这些元素可通过 jQuery 选择器或 DOM 元素来指定。

注释：如果未找到元素，index() 将返回 -1。

第一个匹配元素的 index，相对于同胞元素

获得第一个匹配元素相对于其同胞元素的 index 位置。

语法

```
$(_selector_).index()
```

元素的 index，相对于选择器

获得元素相对于选择器的 index 位置。

该元素可以通过 DOM 元素或 jQuery 选择器来指定。

语法

```
$(_selector_).index(_element_)
```

参数	描述
element	可选。规定要获得 index 位置的元素。可以是 DOM 元素或 jQuery 选择器。

jQuery DOM 元素方法 - size() 方法

实例

输出被 jQuery 选择器匹配的元素的数量：

```
$("button").click(function(){
  alert($("li").size());
});
```

定义和用法

size() 方法返回被 jQuery 选择器匹配的元素的数量。

语法

```
$(_selector_).size()
```

jQuery DOM 元素方法 - toArray() 方法

实例

将 li 元素转换为数组，然后输出该数组元素的 innerHTML ：

```
$("button").click(function(){
  x=$("li").toArray()
  for (i=0;i<x.length;i++)
    {
    alert(x[i].innerHTML);
    }
});
```

定义和用法

toArray() 方法以数组的形式返回 jQuery 选择器匹配的元素。

语法

```
$(_selector_).toArray()
```

jQuery 参考手册 - 核心

jQuery 核心函数

函数	描述
jQuery()	接受一个字符串，其中包含了用于匹配元素集合的 CSS 选择器。
jQuery.noConflict()	运行这个函数将变量 $ 的控制权让渡给第一个实现它的那个库。

jQuery 核心 - jQuery() 方法

实例

找出所有属于 div 元素的子元素的 p 元素，然后设置其边框属性：

```
$("div > p").css("border", "1px solid gray");
```

定义和用法

jQuery() 方法接受一个字符串，其中包含了用于匹配元素集合的 CSS 选择器。

jQuery() 函数有三种语法：

语法 1

接受一个字符串，其中包含了用于匹配元素集合的 CSS 选择器：

```
jQuery(_selector_, [_context_])
```

详细用法

语法 2

使用原始 HTML 的字符串来创建 DOM 元素：

```
jQuery(_html_,[_ownerDocument_])
```

详细用法

语法 3

绑定一个在 DOM 文档载入完成后执行的函数：

```
jQuery( callback )
```

详细用法

jQuery(selector, [context])

该语法有以下几种用法：

用法 1 ：设置选择器环境

语法

```
jQuery(_selector_, [_context_])
```

默认情况下，选择器从文档根部对 DOM 进行搜索。不过，可以为 $() 设置可选的 context 参数。

例如，如果我们希望在一个 callback 中搜索一个元素，可以限定下面的搜索：

实例

```
$("div.foo").click(function() {
  $("span", this).addClass("bar");
});
```

由于我们已经将 span 选择器限定到 this 这个环境中，只有被点击元素中的 span 会得到附加的 class。

在内部，选择器环境是通过 .find() 方法实现的，因此 $("span", this) 等价于 $(this).find("span")。

jQuery 的核心功能都是通过这个函数实现的。jQuery中的一切都基于这个函数，或者说都是在以某种方式使用这个函数。这个函数最基本的用法就是向它传递一个表达式（通常由 CSS 选择器组成），然后根据这个表达式来查找所有匹配的元素。

默认情况下, 如果没有指定 context 参数，$() 将在当前的 HTML document 中查找 DOM 元素；如果指定了 context 参数，如一个 DOM 元素集或 jQuery 对象，那就会在这个 context 中查找。在 jQuery 1.3.2 以后，其返回的元素顺序等同于在 context 中出现的先后顺序。

用法 2 ：使用 DOM 元素

语法

```
jQuery(_element_)
```

该函数允许我们通过使用以其他方式找到的 DOM 元素来创建 jQuery 对象。该功能通常的用法是，对已经通过 this 关键字传递到 callback 函数的元素调用 jQuery 的方法：

实例

```
$("div.foo").click(function() {
  $(this).slideUp();
});
```

此例会在元素被点击时使用滑动动画对其进行隐藏。由于处理程序接受的 this 关键词中的被点击项目是纯的 DOM 元素，因此在对其调用 jQuery 的方法之前，必须用 jQuery 对象包装该元素。

这个函数也可以接收 XML 文档和 Window 对象（虽然它们不是 DOM 元素）作为有效的参数。

当 XML 数据从 Ajax 调用中返回后，我们可以使用 $() 函数通过 jQuery 对象包装该数据。一旦完成，我们就可以使用 .find() 和其他 DOM 遍历方法来取回 XML 结构中单个元素。

用法 3 ：克隆 jQuery 对象

语法

```
jQuery(_jQuery object_)
```

当以参数的形式向 $() 函数传递 jQuery 对象后，会创建一个该对象的副本。与初始对象一样，新的 jQuery 对象引用相同的 DOM 元素。

用法 4 ：返回空的集合

语法

```
jQuery()
```

对于 jQuery 1.4，调用无参数的 jQuery() 方法会返回空的 jQuery 集合。在之前版本的 jQuery 中，这样会返回包含 document 节点的集合。

jQuery(html, [ownerDocument])

该语法有以下几种用法：

用法 1 ：创建新的元素

语法

```
jQuery(_html_,[_ownerDocument_])
```

你可以传递一个手写的 HTML 字符串，或者由某些模板引擎或插件创建的字符串，也可以是通过 AJAX 加载过来的字符串。但是在你创建 input 元素的时会有限制，可以参考第二个示例。

当然这个字符串可以包含斜杠 (比如一个图像地址)，还有反斜杠。当你创建单个元素时，请使用闭合标签或 XHTML 格式。例如，创建一个 span ，可以用 $("<span/>") 或 $("") ，但不推荐 $("<span>")。在 jQuery 中，这个语法等同于 $(document.createElement("span"))。

如果以参数的形式将字符串传递给 $()，jQuery 会检查字符串是否是 HTML （比如，字符串某些位置存在标签）。如果不是，则把字符串解释为选择器表达式，请见上面的讲解。但如果字符串是 HTML 片段，则 jQuery 试图创建由该 HTML 片段描述的 DOM 元素。然后会创建并返回一个引用这些 DOM 元素的 jQuery 对象：

实例

```
$("<p id="test">My <em>new</em> text</p>").appendTo("body");
```

如果 HTML 片段比不含属性的简单标签更复杂，如同上面例子中的 HTML，那么元素实际的创建过程是由浏览器的 innerHTML 机制完成的。具体地讲，jQuery 会创建新的 <div> 元素，然后为传入的 HTML 片段设置元素的 innerHTML 属性。当参数只是简单的标签，比如$("<img />") 或 $("<a>")，jQuery 会通过内生的 JavaScript createElement() 函数来创建元素。

要确保跨平台兼容性，片段的结构必须良好。能够包含其他元素的标签必须成对出现（带有关闭标签）：

```
$("<a href="http://jquery.com"></a>");
```

不过，jQuery 也允许类似 XML 的标签语法：

```
$("<a/>");
```

无法包含其他元素的标签可以关闭，也可以不关闭：

```
$("<img />");
$("<input>");
```

用法 2 ：设置属性和事件

语法

```
jQuery(_html_,_props_)
```

对于 jQuery 1.4，我们可以向第二个参数传递一个属性映射。该参数接受能够传递给 .attr() 方法的属性的超集。此外，可以传递任意的事件类型，并可以调用下面的 jQuery 方法：val, css, html, text, data, width, height, or offset.

注意，Internet Explorer 不允许你创建 input 元素并改变其类型；您必须使用例如 " <input type="checkbox" />" 来规定类型。

实例

创建一个 <input> 元素，同时设定 type 属性、属性值，以及一些事件。

```
$("<input>", {
  type: "text",
  val: "Test",
  focusin: function() {
    $(this).addClass("active");
  },
  focusout: function() {
    $(this).removeClass("active");
  }
}).appendTo("form");
```

jQuery(callback)

允许你绑定一个在 DOM 文档载入完成后执行的函数。

该函数的作用如同 $(document).ready() 一样，只不过用这个函数时，需要把页面中所有需要在 DOM 加载完成时执行的其他 $() 操作符都包装到其中来。尽管从技术上来说，这个函数是可链接的，但真正以这种方式链接的情况并不多。

例子

当DOM加载完成后，执行其中的函数：

```
$(function(){
  // 文档就绪
});
```

jQuery 核心 - noConflict() 方法

实例

使用 noConflict() 方法为 jQuery 变量规定新的名称：

```
var jq=$.noConflict();
```

定义和用法

noConflict() 方法让渡变量 $ 的 jQuery 控制权。

该方法释放 jQuery 对 $ 变量的控制。

该方法也可用于为 jQuery 变量规定新的自定义名称。

提示：在其他 JavaScript 库为其函数使用 $ 时，该方法很有用。

语法

```
jQuery.noConflict(_removeAll_)
```

参数	描述
removeAll	布尔值。指示是否允许彻底将 jQuery 变量还原。

说明

许多 JavaScript 库使用 $ 作为函数或变量名，jQuery 也一样。在 jQuery 中，$ 仅仅是 jQuery 的别名，因此即使不使用 $ 也能保证所有功能性。假如我们需要使用 jQuery 之外的另一 JavaScript 库，我们可以通过调用 $.noConflict() 向该库返回控制权：

```
<script type="text/javascript" src="other_lib.js"></script>
<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">
  $.noConflict();
  // 使用另一个库的 $ 的代码
</script>
```

可以与 .ready() 方法结合起来使用，来为 jQuery 对象起别名，这项技术非常有效：

```
<script type="text/javascript" src="other_lib.js"></script>
<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">
  $.noConflict();
  jQuery(document).ready(function($) {
    // 使用 jQuery $ 的代码
  });
  // 使用其他库的 $ 的代码
</script>
```

此外，通过向该方法传递参数 true，我们可以将 $ 和 jQuery 的控制权都交还给原来的库。用之前请考虑清楚！

这是相对于简单的 noConflict 方法更极端的版本，因为这将完全重新定义 jQuery。这通常用于一种极端的情况，比如你想要将 jQuery 嵌入一个高度冲突的环境。注意：调用此方法后极有可能导致插件失效。

实例

例子 1

将 $ 引用的对象映射回原始的对象：

```
jQuery.noConflict();

jQuery("div p").hide();    // 使用 jQuery

$("content").style.display = "none";    // 使用其他库的 $()
```

例子 2

恢复使用别名 $，然后创建并执行一个函数，在这个函数的作用域中仍然将 $ 作为 jQuery 的别名来使用。在这个函数中，原来的 $ 对象是无效的。这个函数对于大多数不依赖于其他库的插件都十分有效：

```
jQuery.noConflict();

(function($) {
  $(function() {
    // 使用 $ 作为 jQuery 别名的代码
  });
})(jQuery);

... // 其他用 $ 作为别名的库的代码
```

例子 3

可以将 jQuery.noConflict() 与简写的 ready 结合，使代码更紧凑：

```
jQuery.noConflict()(function(){
    // 使用 jQuery 的代码
});

... // 其他库使用 $ 做别名的代码
```

例子 4

创建一个新的别名用以在接下来的库中使用 jQuery 对象：

```
var j = jQuery.noConflict();

j("div p").hide();    // 基于 jQuery 的代码

$("content").style.display = "none";    // 基于其他库的 $() 代码
```

例子 5

完全将 jQuery 移到一个新的命名空间：

```
var dom = {};
dom.query = jQuery.noConflict(true);
```

结果:

```
dom.query("div p").hide();    // 新 jQuery 的代码

$("content").style.display = "none";    // 另一个库 $() 的代码

jQuery("div > p").hide();    // 另一个版本 jQuery 的代码
```

jQuery 参考手册 - 属性

jQuery 属性

下面列出的这些方法设置或返回元素的 CSS 相关属性。

属性	描述
context	在版本 1.10 中被弃用。包含传递给 jQuery() 的原始上下文。
jquery	包含 jQuery 版本号。
jQuery.fx.interval	改变以毫秒计的动画速率。
jQuery.fx.off	全局禁用/启用所有动画。
jQuery.support	表示不同浏览器特性或漏洞的属性集合（用于 jQuery 内部使用）。
length	包含 jQuery 对象中的元素数目。

jQuery context 属性

实例

检测上下文：

```
$("div").append("<p>" + $("div").context + "</p>")
.append("<p>" + $("div",document.body).context.nodeName + "</p>");
```

定义和用法

context 属性在 jQuery version 1.10 中被弃用。

context 属性含有被传递到 jQuery 的原始上下文，可能是 DOM 节点上下文，如果未传递节点，则是 document 上下文。

语法

```
context
```

jQuery jquery 属性

实例

输出当前正在运行的 jQuery 版本：

```
$("button").on("click",function(){
  var version = $().jquery;
  alert("You are running jQuery version: " + version);
});
```

定义和用法

jquery 属性返回的字符串包含 jquery 的版本号。

语法

```
$().jquery
```

jQuery jQuery.fx.interval 属性

实例

以较少的帧数来运行 <div> 元素的动画：

```
$("#toggle").on("click",function(){
  $("div").toggle(5000);
});
$("#interval").on("click",function(){
  jQuery.fx.interval = 500;
});
```

定义和用法

jQuery.fx.interval 属性用于改变以毫秒计的动画运行速率。可操作该属性来调整动画运行的每秒帧数。

默认值是 13 毫秒。该属性常用于修改动画运行的每秒帧数。

降低这个值能够使动画在更快的浏览器中运行得更流畅，但这么做也行会影响性能。

提示：由于 jQuery 使用一个全局的间隔时间，为了使该属性生效，动画应该不在运行或者首先停止所有动画。

注释：该属性在支持 requestAnimationFrame 属性的浏览器中无效，比如 Google Chrome 11。

语法

```
jQuery.fx.interval = _milliseconds_;
```

属性	描述
milliseconds	必需。规定以毫秒计的动画运行速率。默认是 13 毫秒。

jQuery jQuery.fx.off 属性

实例

切换动画开关：

```
$("#true").click(function(){
  jQuery.fx.off = true;
});
$("#false").click(function(){
  jQuery.fx.off = false;
});
$("#toggle").click(function(){
  $("div").toggle("slow");
});
```

定义和用法

jQuery.fx.off 属性用于对所有动画进行全局禁用或启用。

默认值是 false，它允许动画正常运行。当设置为 true 时，将禁用所有动画方法，这样会把元素设置为其最后的状态，而不是显示效果。

提示：如需简化代码，可以使用 $.fx.off 来代替 jQuery.fx.off。

语法

```
jQuery.fx.off = true|false;
```

属性	描述
true	规定应该禁用动画。
false	默认。规定应该启用动画。

jQuery jQuery.support 属性

实例

测试浏览器是否能创建 XMLHttpRequest 对象：

```
$(document).ready(function(){
  $("p").html("This browser can create XMLHttpRequest object: " + j
});
```

定义和用法

jQuery.support 属性包含表示不同浏览器特性或漏洞的属性集。

此属性主要用于 jQuery 的内部使用。

语法

```
jQuery.support._propvalue_
```

属性	描述
propvalue	必需。规定要测试的功能。这些测试包括： `ajax` `boxModel` `changeBubbles` `checkClone` `checkOn` `cors` `cssFloat` `hrefNormalized` `htmlSerialize` `leadingWhitespace` `noCloneChecked` `noCloneEvent` `opacity` `optDisabled` `optSelected` `scriptEval()` `style` `submitBubbles` `tbody`

jQuery length 属性

实例

输出 <li> 元素的数目：

```
$("button").click(function(){
  alert($("li").length);
});
```

定义和用法

length 属性包含 jQuery 对象中元素的数目。

语法

```
$(selector).length
```

W3School AngularJS教程

来源：AngularJS 教程

整理：飞龙

AngularJS 简介

AngularJS 是一个 **JavaScript** 框架。它可通过 <script> 标签添加到 HTML 页面。

AngularJS 通过 指令 扩展了 HTML，且通过 表达式 绑定数据到 HTML。

AngularJS 是一个 JavaScript 框架

AngularJS 是一个 JavaScript 框架。它是一个以 JavaScript 编写的库。

AngularJS 是以一个 JavaScript 文件形式发布的，可通过 script 标签添加到网页中：

```
<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js"
```

> 我们建议把脚本放在 <body> 元素的底部。
> 这会提高网页加载速度，因为 HTML 加载不受制于脚本加载

AngularJS 扩展了 HTML

AngularJS 通过 **ng-directives** 扩展了 HTML。

ng-app 指令定义一个 AngularJS 应用程序。

ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

ng-bind 指令把应用程序数据绑定到 HTML 视图。

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="">
  <p>在输入框中尝试输入：</p>
  <p>姓名：<input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

</body>
</html>
```

实例讲解：

当网页加载完毕，AngularJS 自动开启。

ng-app 指令告诉 AngularJS，<div> 元素是 AngularJS 应用程序 的"所有者"。

ng-model 指令把输入域的值绑定到应用程序变量 **name**。

ng-bind 指令把应用程序变量 name 绑定到某个段落的 innerHTML。

> 如果您移除了 **ng-app** 指令，HTML 将直接把表达式显示出来，不会去计算表达式的结果。

什么是 **AngularJS**？

"AngularJS 是专门为应用程序设计的 HTML。"

AngularJS 使得开发现代的单一页面应用程序（SPAs：Single Page Applications）变得更加容易。

- AngularJS 把应用程序数据绑定到 HTML 元素。
- AngularJS 可以克隆和重复 HTML 元素。
- AngularJS 可以隐藏和显示 HTML 元素。
- AngularJS 可以在 HTML 元素"背后"添加代码。
- AngularJS 支持输入验证。

AngularJS 指令

正如您所看到的，AngularJS 指令是以 **ng** 作为前缀的 HTML 属性。

ng-init 指令初始化 AngularJS 应用程序变量。

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John'">

<p>姓名为 <span ng-bind="firstName"></span></p>

</div>
```

> HTML5 允许扩展的（自制的）属性，以 **data-** 开头。
> AngularJS 属性以 **ng-** 开头，但是您可以使用 **data-ng-** 来让网页对 HTML5 有效。

带有有效的 HTML5：

AngularJS 实例

```
<div data-ng-app="" data-ng-init="firstName='John'">

<p>姓名为 <span data-ng-bind="firstName"></span></p>

</div>
```

AngularJS 表达式

AngularJS 表达式写在双大括号内：**。

AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 **JavaScript** 表达式：它们可以包含文字、运算符和变量。

实例 10 或 undefined undefined

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="">
  <p>我的第一个表达式： {{ 5 + 5 }}</p>
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

</body>
</html>
```

AngularJS 表达式

AngularJS 使用 表达式 把数据绑定到 HTML。

AngularJS 表达式

AngularJS 表达式写在双大括号内：**。

AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 **JavaScript** 表达式：它们可以包含文字、运算符和变量。

实例 10 或 undefined undefined

AngularJS 数字

AngularJS 数据就像 JavaScript 数字：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： {{ quantity * cost }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： <span ng-bind="quantity * cost"></span></p>

</div>
```

> 使用 **ng-init** 不是很常见。您将在控制器一章中学习到一个更好的初始化数据的方式。

AngularJS 字符串

AngularJS 字符串就像 JavaScript 字符串：

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： {{ firstName + " " + lastName }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： <span ng-bind="firstName + ' ' + lastName"></span></p>

</div>
```

AngularJS 对象

AngularJS 对象就像 JavaScript 对象：

AngularJS 实例

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 {{ person.lastName }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 <span ng-bind="person.lastName"></span></p>

</div>
```

AngularJS 数组

AngularJS 数组就像 JavaScript 数组：

AngularJS 实例

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 {{ points[2] }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 <span ng-bind="points[2]"></span></p>

</div>
```

AngularJS 指令

AngularJS 通过被称为 指令 的新属性来扩展 HTML。

AngularJS 指令

AngularJS 指令是扩展的 HTML 属性，带有前缀 **ng-**。

ng-app 指令初始化一个 AngularJS 应用程序。

ng-init 指令初始化应用程序数据。

ng-model 指令把应用程序数据绑定到 HTML 元素。

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John'">

  <p>在输入框中尝试输入：</p>
  <p>姓名：<input type="text" ng-model="firstName"></p>
  <p>你输入的为： {{ firstName }}</p>

</div>
```

ng-app 指令告诉 AngularJS，<div> 元素是 AngularJS 应用程序 的"所有者"。

> 一个网页可以包含多个运行在不同元素中的 AngularJS 应用程序。

数据绑定

上面实例中的 ** 表达式是一个 AngularJS 数据绑定表达式。

AngularJS 中的数据绑定，同步了 AngularJS 表达式与 AngularJS 数据。

** 是通过 ng-model="firstName"** 进行同步。

在下一个实例中，两个文本域是通过两个 ng-model 指令同步的：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;price=5">

<h2>价格计算器</h2>

数量： <input type="number" ng-model="quantity">
价格： <input type="number" ng-model="price">

<p><b>总价：</b> {{ quantity * price }}</p>

</div>
```

> 使用 **ng-init** 不是很常见。您将在控制器一章中学习到一个更好的初始化数据的方式。

重复 HTML 元素

ng-repeat 指令会重复一个 HTML 元素：

AngularJS 实例

```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
  <p>使用 ng-repeat 来循环数组</p>
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
<div>
```

ng-repeat 指令用在一个对象数组上：

AngularJS 实例

```
<div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">

<p>循环对象：</p>
<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>
```

> AngularJS 完美支持数据库的 CRUD（增加Create、读取Read、更新Update、删除Delete）应用程序。
> 把实例中的对象想象成数据库中的记录。

ng-app 指令

ng-app 指令定义了 AngularJS 应用程序的 根元素。

ng-app 指令在网页加载完毕时会自动引导（自动初始化）应用程序。

稍后您将学习到 **ng-app** 如何通过一个值（比如 ng-app="myModule"）连接到代码模块。

ng-init 指令

ng-init 指令为 AngularJS 应用程序定义了 初始值。

通常情况下，不使用 ng-init。您将使用一个控制器或模块来代替它。

稍后您将学习更多有关控制器和模块的知识。

ng-model 指令

ng-model 指令 绑定 **HTML** 元素 到应用程序数据。

ng-model 指令也可以：

- 为应用程序数据提供类型验证（number、email、required）。
- 为应用程序数据提供状态（invalid、dirty、touched、error）。
- 为 HTML 元素提供 CSS 类。
- 绑定 HTML 元素到 HTML 表单。

ng-repeat 指令

ng-repeat 指令对于集合中（数组中）的每个项会 克隆一次 **HTML** 元素。

AngularJS 控制器

AngularJS 控制器 控制 AngularJS 应用程序的数据。

AngularJS 控制器是常规的 **JavaScript** 对象。

AngularJS 控制器

AngularJS 应用程序被控制器控制。

ng-controller 指令定义了应用程序控制器。

控制器是 **JavaScript** 对象，由标准的 JavaScript 对象的构造函数 创建。

控制器的 **$scope** 是控制器所指向的应用程序/HTML 元素。

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{person.firstName + " " + person.lastName}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
}
</script>
```

实例讲解：

AngularJS 应用程序由 **ng-app** 定义。应用程序在 <div> 内运行。

ng-controller 指令把控制器命名为 **object**。

函数 **personController** 是一个标准的 JavaScript 对象的构造函数。

控制器对象有一个属性：**$scope.person**。

person 对象有两个属性：**firstName** 和 **lastName**。

ng-model 指令绑定输入域到控制器的属性（firstName 和 lastName）。

控制器属性

上面的实例演示了一个带有 lastName 和 firstName 这两个属性的控制器对象。

控制器也可以把函数作为对象属性：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{person.fullName()}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe",
        fullName: function() {
            var x;
            x = $scope.person;
            return x.firstName + " " + x.lastName;
        }
    };
}
</script>
```

控制器方法

控制器也可以带有方法：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{fullName()}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe",
     };
     $scope.fullName = function() {
         var x;
         x = $scope.person;
         return x.firstName + " " + x.lastName;
     };
}
</script>
```

外部文件中的控制器

在大型的应用程序中，通常是把控制器存储在外部文件中。

只需要把 <script> 标签中的代码复制到名为 personController.js 的外部文件中即可：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{person.firstName + " " + person.lastName}}

</div>

<script src="personController.js"></script>
```

另一个实例

下面的实例我们将创建一个新的控制器文件：

```
function namesController($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Hege',country:'Sweden'},
        {name:'Kai',country:'Denmark'}
    ];
}
```

然后在应用程序中使用这个控制器文件：

AngularJS 实例

```
<div ng-app="" ng-controller="namesController">

<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>

<script src="namesController.js"></script>
```

AngularJS 过滤器

过滤器可以使用一个管道字符（|）添加到表达式和指令中。

AngularJS 过滤器

AngularJS 过滤器可用于转换数据：

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

向表达式添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到表达式中。

（下面的两个实例，我们将使用前面章节中提到的 person 控制器）

uppercase 过滤器格式化字符串为大写：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

<p>姓名为 {{ person.lastName | uppercase }}</p>

</div>
```

lowercase 过滤器格式化字符串为小写：

AngularJS 实例

```
<div ng-app="" ng-controller="personController"">

<p>姓名为 {{ person.lastName | lowercase }}</p>

</div>
```

currency 过滤器

currency 过滤器格式化数字为货币格式：

AngularJS 实例

```
<div ng-app="" ng-controller="costController">

数量：<input type="number" ng-model="quantity">
价格：<input type="number" ng-model="price">

<p>总价 = {{ (quantity * price) | currency }}</p>

</div>
```

向指令添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中。

orderBy 过滤器根据某个表达式排列数组：

AngularJS 实例

```
<div ng-app="" ng-controller="namesController">

<p>循环对象：</p>
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

<div>
```

过滤输入

输入过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中，该过滤器后跟一个冒号和一个模型名称。

filter 过滤器从数组中选择一个子集：

AngularJS 实例

```
<div ng-app="" ng-controller="namesController">

<p>输入过滤：</p>
<p><input type="text" ng-model="name"></p>

<ul>
  <li ng-repeat="x in names | filter:name | orderBy:'country'">
    {{ (x.name | uppercase) + ', ' + x.country }}
  </li>
</ul>

</div>
```

AngularJS XMLHttpRequest

$http 是 AngularJS 中的一个核心服务，用于读取远程服务器的数据。

读取 JSON 文件

以下是存储在web服务器上的 JSON 文件：

http://www.w3cschool.cc/try/angularjs/data/Customers_JSON.php


```
[
{
"Name" : "Alfreds Futterkiste",
"City" : "Berlin",
"Country" : "Germany"
},
{
"Name" : "Berglunds snabbk?p",
"City" : "Lule?",
"Country" : "Sweden"
},
{
"Name" : "Centro comercial Moctezuma",
"City" : "México D.F.",
"Country" : "Mexico"
},
{
"Name" : "Ernst Handel",
"City" : "Graz",
"Country" : "Austria"
},
{
"Name" : "FISSA Fabrica Inter. Salchichas S.A.",
"City" : "Madrid",
"Country" : "Spain"
},
{
"Name" : "Galería del gastrónomo",
"City" : "Barcelona",
"Country" : "Spain"
},
{
"Name" : "Island Trading",
"City" : "Cowes",
"Country" : "UK"
},
{
"Name" : "K?niglich Essen",
```

```
"City" : "Brandenburg",
"Country" : "Germany"
},
{
"Name" : "Laughing Bacchus Wine Cellars",
"City" : "Vancouver",
"Country" : "Canada"
},
{
"Name" : "Magazzini Alimentari Riuniti",
"City" : "Bergamo",
"Country" : "Italy"
},
{
"Name" : "North/South",
"City" : "London",
"Country" : "UK"
},
{
"Name" : "Paris spécialités",
"City" : "Paris",
"Country" : "France"
},
{
"Name" : "Rattlesnake Canyon Grocery",
"City" : "Albuquerque",
"Country" : "USA"
},
{
"Name" : "Simons bistro",
"City" : "K?benhavn",
"Country" : "Denmark"
},
{
"Name" : "The Big Cheese",
"City" : "Portland",
"Country" : "USA"
},
{
"Name" : "Vaffeljernet",
"City" : "?rhus",
"Country" : "Denmark"
},
{
"Name" : "Wolski Zajazd",
"City" : "Warszawa",
"Country" : "Poland"
}
]
```


AngularJS $http

AngularJS $http 是一个用于读取web服务器上数据的服务。

$http.get(url) 是用于读取服务器数据的函数。

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<ul>
  <li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>

</div>

<script>
function customersController($scope,$http) {
    $http.get("http://www.w3cschool.cc/try/angularjs/data/Customers
    .success(function(response) {$scope.names = response;});
}
</script>
```

应用解析:

AngularJS 应用通过 **ng-app** 定义。应用在 <div> 中执行。

ng-controller 指令设置了 **controller** 对象 名。

函数 **customersController** 是一个标准的 JavaScript 对象构造器。

控制器对象有一个属性: **$scope.names**。

$http.get() 从web服务器上读取静态 **JSON** 数据。

服务器数据文件为： **http://www.w3cschool.cc/try/angularjs/data/Customers_JSON.php**。

当从服务端载入 JSON 数据时，**$scope.names** 变为一个数组。

> 以上代码也可以用于读取数据库数据。

AngularJS 表格

ng-repeat 指令可以完美的显示表格。

在表格中显示数据

使用 angular 显示表格是非常简单的：

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
function customersController($scope,$http) {
  $http.get("http://www.w3cschool.cc/try/angularjs/data/Customers_J
  .success(function(response) {$scope.names = response;});
}
</script>
```

使用 CSS 样式

为了让页面更加美观，我们可以在页面中使用CSS：

CSS 样式

```
<style>
table, th , td {
  border: 1px solid grey;
  border-collapse: collapse;
  padding: 5px;
}
table tr:nth-child(odd) {
  background-color: #f1f1f1;
}
table tr:nth-child(even) {
  background-color: #ffffff;
}
</style>
```

排序显示

如果需要对表格进行排序，我们可以添加 **orderBy** 过滤器:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names | orderBy : 'Country'">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
```

Try it Yourself ?

使用 uppercase 滤器

如果字母要转换为大写，可以添加 **uppercase** 过滤器:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country | uppercase}}</td>
  </tr>
</table>
```

AngularJS SQL

在前面章节中的代码也可以用于读取数据库中的数据。

使用 PHP 从 MySQL 中获取数据

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
function customersController($scope,$http) {
    var site = "http://www.w3cschool.cc";
    var page = "/try/angularjs/data/Customers_MySQL.php";
    $http.get(site + page)
    .success(function(response) {$scope.names = response;});
}
</script>
```

ASP.NET 中执行 SQL 获取数据

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
function customersController($scope,$http) {
    var site = "http://www.w3cschool.cc";
    var page = "/try/angularjs/data/Customers_SQL.aspx";
    $http.get(site + page)
    .success(function(response) {$scope.names = response;});
}
</script>
```

PHP 读取 MySQL 数据代码

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: text/html; charset=UTF-8");

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind")

$result = $conn->query("SELECT CompanyName, City, Country FROM Cust

$outp "[";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "[") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
}
$outp .="]";

$conn->close();

echo($outp);
?>
```

PHP 读取 MS Access 代码

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: text/html; charset=ISO-8859-1");

$conn = new COM("ADODB.Connection");
$conn->open("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=Northwinc

$rs = $conn->execute("SELECT CompanyName, City, Country FROM Custor

$outp = "[";
while (!$rs->EOF) {
    if ($outp != "[") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
    $rs->MoveNext();
}
$outp .= "]";

$conn->close();

echo ($outp);
?>
```

服务端 ASP.NET, VB 和 MS Access 代码

```
<%@ Import Namespace="System.IO"%>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System.Data.OleDb"%>
<%
Response.AppendHeader("Access-Control-Allow-Origin", "*")
Dim conn As OleDbConnection
Dim objAdapter As OleDbDataAdapter
Dim objTable As DataTable
Dim objRow As DataRow
Dim objDataSet As New DataSet()
Dim outp
Dim c
conn = New OledbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data s
objAdapter = New OledbDataAdapter("SELECT CompanyName, City, Countr
objAdapter.Fill(objDataSet, "myTable")
objTable=objDataSet.Tables("myTable")

outp = "["
c = chr(34)
for each x in objTable.Rows
if outp <> "[" then outp = outp & ","
outp = outp & "{" & c & "Name"    & c & ":" & c & x("CompanyName")
outp = outp &       c & "City"    & c & ":" & c & x("City")
outp = outp &       c & "Country" & c & ":" & c & x("Country")
next

outp = outp & "]"
response.write(outp)
conn.close
%>
```

服务端 ASP.NET, VB Razor 和 SQL Lite 代码

```
@{
Response.AppendHeader("Access-Control-Allow-Origin", "*")
var db = Database.Open("Northwind");
var query = db.Query("SELECT CompanyName, City, Country FROM Custor
var outp ="["
}
@foreach(var row in query)
{
if outp <> "[" then outp = outp + ","
outp = outp + "{" + c + "Name"    + c + ":" + c + @row.CompanyName
outp = outp +       c + "City"    + c + ":" + c + @row.City
outp = outp +       c + "Country" + c + ":" + c + @row.Country
}
outp = outp + "]"
@outp
```

AngularJS HTML DOM

AngularJS 有自己的 HTML 属性指令。

ng-disabled 指令

ng-disabled 指令直接绑定应用程序数据到 HTML 的 disabled 属性。

AngularJS 实例

```
<div ng-app="">

<p>
<button ng-disabled="mySwitch">点我！</button>
</p>

<p>
<input type="checkbox" ng-model="mySwitch">按钮
</p>

</div>
```

实例讲解：

ng-disabled 指令绑定应用程序数据 "mySwitch" 到 HTML 的 disabled 属性。

ng-model 指令绑定 "mySwitch" 到 HTML input checkbox 元素的内容（value）。

ng-show 指令

ng-show 指令隐藏或显示一个 HTML 元素。

AngularJS 实例

```
<div ng-app="">

<p ng-show="true">我是可见的。</p>

<p ng-show="false">我是不可见的。</p>

</div>
```

您可以使用一个评估为 true or false 的表达式（比如 ng-show="hour < 12"）来隐藏和显示 HTML 元素。

在下一章中，有另一个实例，通过单击一个按钮来隐藏一个 HTML 元素。

AngularJS HTML 事件

AngularJS 有自己的 HTML 事件指令。

ng-click 指令

ng-click 指令定义了一个 AngularJS 单击事件。

AngularJS 实例

```
<div ng-app="" ng-controller="myController">

<button ng-click="count = count + 1">点我！</button>

<p>{{ count }}</p>

</div>
```

隐藏 HTML 元素

ng-hide 指令用于设置应用的一部分 不可见 。

ng-hide="true" 让 HTML 元素 不可见。

ng-hide="false" 让元素可见。

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

<button ng-click="toggle()">隐藏/显示</button>

<p ng-hide="myVar">
名: <input type="text" ng-model="person.firstName"><br>
姓: <input type="text" ng-model="person.lastName"><br>
<br>
姓名: {{person.firstName + " " + person.lastName}}
</p>

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
    $scope.myVar = false;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    };
}
</script>
```

应用解析:

personController的第一部分与控制器章节类似。

应用有一个默认属性: **$scope.myVar = false;**

ng-hide 指令设置应用中的元素不可见。

toggle() 函数用于切换 **myVar** 变量的值（true 和 false）。

ng-hide="true" 让元素 不可见。

显示 HTML 元素

ng-show 指令可用于设置应用中的一部分可见 。

ng-show="false" 可以设置 HTML 元素 不可见。

ng-show="true" 可以以设置 HTML 元素可见。

以下实例使用了 ng-show 指令:

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

<button ng-click="toggle()">隐藏/显示</button>

<p ng-show="myVar">
名: <input type="text" ng-model="person.firstName"><br>
姓: <input type="text" ng-model="person.lastName"><br>
<br>
姓名: {{person.firstName + " " + person.lastName}}
</p>

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
    $scope.myVar = true;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    };
}
</script>
```

AngularJS 模块

模块定义了您的应用程序。

所有的控制器都应该属于一个模块。

模块保持全局命名空间中的整洁。

AngularJS 模块实例

在本实例中，"myApp.js" 包含了一个应用程序模块定义，"myCtrl.js" 包含了一个控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

控制器污染了全局命名空间

本教程中，截至目前为止的所有实例都使用了全局函数。

在所有的应用程序中，都应该尽量避免使用全局变量和全局函数。

全局值（变量或函数）可被其他脚本重写或破坏。

为了解决这个问题，AngularJS 使用了模块。

AngularJS 模块

使用一个简单的 控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
function myCtrl($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
}
</script>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

</body>
</html>
```

使用一个由 模块 替代的控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'
</head>

<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>

</body>
</html>
```

> 请注意，本实例中，AngularJS 库是在 <head> 区域被加载。

模块定义应放置在何处？

对于 HTML 应用程序，通常建议把所有的脚本都放置在 <body> 元素的最底部。

这会提高网页加载速度，因为 HTML 加载不受制于脚本加载。

在上面的多个 AngularJS 实例中，您将看到 AngularJS 库是在文档的 <head> 区域被加载。

在上面的实例中，AngularJS 在 <head> 元素中被加载，因为对 angular.module 的调用只能在库加载完成后才能进行。

另一个解决方案是在 <body> 元素中加载 AngularJS 库，但是必须放置在您的 AngularJS 脚本前面：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

<script>
var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>

</body>
</html>
```

AngularJS 应用程序文件

现在您已经知道模块是什么以及它们是如何工作的，现在您可以尝试创建您自己的应用程序文件。

您的应用程序至少应该有一个模块文件，一个控制器文件。

首先，创建模块文件 "myApp.js"：

```
var app = angular.module("myApp", []);
```

然后，创建控制器文件。本实例中是 "myCtrl.js"：

```
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
```

最后，编辑您的 HTML 页面：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

AngularJS 表单

AngularJS 表单是输入控件的集合。

HTML 控件

以下 HTML input 元素被称为 HTML 控件:

- input 元素
- select 元素
- button 元素
- textarea 元素

HTML 表单

HTML 表单通常与 HTML 控件同时存在。

AngularJS 表单实例

```
<script>function ExampleController($scope) { $scope.master = {"firs

<form novalidate="" class="ng-pristine ng-valid">First Name:
<input type="text" ng-model="user.firstName" class="ng-pristine ng-
Last Name:
<input type="text" ng-model="user.lastName" class="ng-pristine ng-v

<button ng-click="reset()">RESET</button></form>

form = {"firstName":"John","lastName":"Doe"}

master = {"firstName":"John","lastName":"Doe"}
```

应用程序代码

```
<div ng-app="" ng-controller="formController">
  <form novalidate>
    First Name:<br>
    <input type="text" ng-model="user.firstName"><br>
    Last Name:<br>
    <input type="text" ng-model="user.lastName">
    <br><br>
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user}}</p>
  <p>master = {{master}}</p>
</div>

<script>
function formController ($scope) {
    $scope.master = {firstName: "John", lastName: "Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
};
</script>
```

HTML 属性 **novalidate** 用于禁用浏览器的默认验证。

实例解析

AngularJS **ng-model** 指令用于绑定 input 元素到模型中。

模型对象 **master** 的值为 {"firstName" : "John", "lastName" : "Doe"}。

模型函数 **reset** 设置了模型对象 **user** 等于 master。

AngularJS 输入验证

AngularJS 表单和控件可以验证输入的数据。

输入验证

在前面的几个章节中，你已经学到关于 AngularJS 表单和控件的知识。

AngularJS 表单和控件可以提供验证功能，并对用户输入的非法数据进行警告。

> 客户端的验证不能确保用户输入数据的安全，所以服务端的数据验证也是必须的。

应用代码

```
<!DOCTYPE html>
<html>

<body>
<h2>Validation Example</h2>

<form  ng-app=""  ng-controller="validateCtrl"
name="myForm" novalidate>

<p>Username:<br>
  <input type="text" name="user" ng-model="user" required>
  <span style="color:red" ng-show="myForm.user.$dirty && myForm.use
  <span ng-show="myForm.user.$error.required">Username is required
  </span>
</p>

<p>Email:<br>
  <input type="email" name="email" ng-model="email" required>
  <span style="color:red" ng-show="myForm.email.$dirty && myForm.er
  <span ng-show="myForm.email.$error.required">Email is required.</
  <span ng-show="myForm.email.$error.email">Invalid email address.<
  </span>
</p>

<p>
  <input type="submit"
  ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||
  myForm.email.$dirty && myForm.email.$invalid">
</p>

</form>

<script src="//apps.bdimg.com/libs/angular.js/1.2.15/angular.min.js
<script>
function validateCtrl($scope) {
    $scope.user = 'John Doe';
    $scope.email = 'john.doe@gmail.com';
}
</script>

</body>
</html>
```

HTML 表单属性 **novalidate** 用于禁用浏览器默认的验证。

实例解析

AngularJS **ng-model** 指令用于绑定输入元素到模型中。

模型对象有两个属性： **user** 和 **email**。

我们使用了 **ng-show**指令， color:red 在邮件是 **$dirty** 或 **$invalid** 才显示。

AngularJS Bootstrap

AngularJS 的首选样式表是 Twitter Bootstrap， Twitter Bootstrap 是目前最受欢迎的前端框架。

查看 Bootstrap教程。

Bootstrap

你可以在你的 AngularJS 应用中加入 Twitter Bootstrap，你可以在你的 <head>元素中添加如下代码:

```
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3
```

如果站点在国内，建议使用百度静态资源库的Bootstrap，代码如下：

```
<link rel="stylesheet" href="//apps.bdimg.com/libs/bootstrap/3.2.0/
```

以下是一个完整的 HTML 实例, 使用了 AngularJS 指令和 Bootstrap 类。

HTML 代码

```
<!DOCTYPE html>
<html ang-app="">
<head>
<link rel="stylesheet" href="http://apps.bdimg.com/libs/bootstrap/3
</head>

<body ng-controller="userController">
<div class="container">

<h3>Users</h3>

<table class="table table-striped">
  <thead><tr>
    <th>Edit</th>
    <th>First Name</th>
    <th>Last Name</th>
  </tr></thead>
  <tbody><tr ng-repeat="user in users">
    <td>
```

```
      <button class="btn" ng-click="editUser(user.id)">
      <span class="glyphicon glyphicon-pencil"></span>  E
      </button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
  </tr></tbody>
</table>

<hr>
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1" placeholder="Password"
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2" placeholder="Repeat Pa
  </div>
</div>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
</div>

<script src = "http://apps.bdimg.com/libs/angular.js/1.2.15/angula
<script src = "myUsers.js"></script>
```

```
</body>
</html>>
```

指令解析

AngularJS 指令	描述
html ng-app	为 `<html>` 元素定义一个应用(未命名)
body ng-controller	为 `<body>` 元素定义一个控制器
tr ng-repeat	循环 users 对象数组，每个 user 对象放在 `<tr>` 元素中。
button ng-click	当点击 `<button>` 元素时调用函数 editUser()
h3 ng-show	如果 edit = true 显示 `<h3>` 元素
h3 ng-hide	如果 edit = true 隐藏 `<h3>` 元素
input ng-model	为应用程序绑定 `<input>` 元素
button ng-disabled	如果发生错误或者 ncomplete = true 禁用 `<button>` 元素

Bootstrap 类解析

元素	Bootstrap 类	定义
`<div>`	container	内容容器
`<table>`	table	表格
`<table>`	table-striped	带条纹背景的表格
`<button>`	btn	按钮
`<button>`	btn-success	成功按钮
`<span>`	glyphicon	字形图标
`<span>`	glyphicon-pencil	铅笔图标
`<span>`	glyphicon-user	用户图标
`<span>`	glyphicon-save	保存图标
`<form>`	form-horizontal	水平表格
`<div>`	form-group	表单组
`<label>`	control-label	控制器标签
`<label>`	col-sm-2	跨越 2 列
`<div>`	col-sm-10	跨越 10 列

JavaScript 代码

```
function userController($scope) {
$scope.fName = '';
$scope.lName = '';
$scope.passw1 = '';
$scope.passw2 = '';
$scope.users = [
{id:1, fName:'Hege',  lName:"Pege" },
{id:2, fName:'Kim',   lName:"Pim" },
{id:3, fName:'Sal',   lName:"Smith" },
{id:4, fName:'Jack',  lName:"Jones" },
{id:5, fName:'John',  lName:"Doe" },
{id:6, fName:'Peter', lName:"Pan" }
];
$scope.edit = true;
$scope.error = false;
$scope.incomplete = false;

$scope.editUser = function(id) {
  if (id == 'new') {
    $scope.edit = true;
    $scope.incomplete = true;
    $scope.fName = '';
    $scope.lName = '';
    } else {
    $scope.edit = false;
    $scope.fName = $scope.users[id-1].fName;
    $scope.lName = $scope.users[id-1].lName;
  }
};

$scope.$watch('passw1',function() {$scope.test();});
$scope.$watch('passw2',function() {$scope.test();});
$scope.$watch('fName', function() {$scope.test();});
$scope.$watch('lName', function() {$scope.test();});

$scope.test = function() {
  if ($scope.passw1 !== $scope.passw2) {
    $scope.error = true;
    } else {
    $scope.error = false;
  }
  $scope.incomplete = false;
  if ($scope.edit && (!$scope.fName.length ||
  !$scope.lName.length ||
  !$scope.passw1.length || !$scope.passw2.length)) {
       $scope.incomplete = true;
  }
};

}
```

JavaScript 代码解析

Scope 属性	用途
$scope.fName	模型变量 (用户名)
$scope.lName	模型变量 (用户姓)
$scope.passw1	模型变量 (用户密码 1)
$scope.passw2	模型变量 (用户密码 2)
$scope.users	模型变量 (用户的数组)
$scope.edit	当用户点击创建用户时设置为true。
$scope.error	如果 passw1 不等于 passw2 设置为 true
$scope.incomplete	如果每个字段都为空(length = 0)设置为 true
$scope.editUser	设置模型变量
$scope.watch	监控模型变量
$scope.test	验证模型变量的错误和完整性

AngularJS Include（包含）

使用 AngularJS, 你可以在 HTML 中包含 HTML 文件。

在未来的HTML中包含 HTML 文件

在 HTML 中，目前还不支持包含 HTML 文件的功能。

W3C 已经建议 http://www.w3.org 在未来的 HTML 中支持包含HTML的功能，格式如下:

```
<link rel="import" href="/path/navigation.html">
```

服务端包含

大部分web服务器支持服务端脚本的包含 (**SSI：Server Side Includes**)。

使用 SSI, 你可以在HTML页面发送至浏览器前包含 HTML。

PHP 实例

```
<?php require("navigation.php"); ?>
```

客户端包含

客户端在 HTML 中使用 JavaScript 有多种方式可以包含 HTML 文件。

通常我们使用 http 请求 (**AJAX**) 从服务端获取数据，返回的数据我们可以通过 使用 **innerHTML** 写入到 HTML 元素中。

AngularJS 包含

使用 AngularJS, 你可以使用 **ng-include** 指令来包含 HTML 内容:

实例

```
<body>

<div class="container">
  <div ng-include="'myUsers_List.htm'"></div>
  <div ng-include="'myUsers_Form.htm'"></div>
</div>

</body>
```

步骤如下。

步骤 1: 创建 HTML 列表

myUsers_List.html

```
<table class="table table-striped">
  <thead><tr>
    <th>Edit</th>
    <th>First Name</th>
    <th>Last Name</th>
  </tr></thead>
  <tbody><tr ng-repeat="user in users">
    <td>
      <button class="btn" ng-click="editUser(user.id)">
        <span class="glyphicon glyphicon-pencil"></span>  
      </button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
  </tr></tbody>
</table>
```

步骤 2: 创建 HTML 表单

myUsers_List.html

```
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1" placeholder="Password'
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2" placeholder="Repeat Pa
  </div>
</div>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
```

步骤 3: 创建主页

myUsers.html

```
<!DOCTYPE html>
<html ng-app="">
<head>
<link rel="stylesheet" href = "http://apps.bdimg.com/libs/bootstrap
</head>

<body ng-controller="userController">

<div class="container">
<div ng-include="'myUsers_List.htm'"></div>
<div ng-include="'myUsers_Form.htm'"></div>
</div>

<script src= "http://apps.bdimg.com/libs/angular.js/1.2.15/angular.
<script src= "myUsers.js"></script>

</body>
</html>
```

AngularJS 应用程序

现在是时候创建一个真正的 AngularJS 应用程序了。

AngularJS 应用程序

您已经学习了足够多关于 AngularJS 的知识，现在可以开始创建您的第一个 AngularJS 应用程序：

我的笔记

```
<textarea cols="40" rows="10"></textarea>

<button>保存</button> <button>清除</button>

剩下的字符数：
```

应用程序讲解

AngularJS 实例

```
<div ng-app="myTodoApp" ng-controller="myTodoCtrl">

<h2>我的笔记</h2>

<p><textarea ng-model="message" cols="40" rows="10"></textarea></p>

<p>
<button ng-click="save()">保存</button>
<button ng-click="clear()">清除</button>
</p>

<p>剩下的字符数：<span ng-bind="left()"></span></p>

</div>

<script src="myTodoApp.js"></script>
<script src="myTodoCtrl.js"></script>
```

应用程序文件 "myTodoApp.js"：

```
var app = angular.module("myTodoApp", []);
```

控制器文件 "myTodoCtrl.js"：

```
app.controller("myTodoCtrl", function($scope) {
    $scope.message = "";
    $scope.left  = function() {return 100 - $scope.message.length;]
    $scope.clear = function() {$scope.message="";};
    $scope.save  = function() {$scope.message="";};
});
```

HTML 页面中的一个 <div>，指向 ng-app="**myTodoApp**" 和 ng-controller="**myTodoCtrl**"：

```
<div ng-app="myTodoApp" ng-controller="myTodoCtrl">
```

一个 **ng-model** 指令，绑定一个 <textarea> 到控制器变量 **message**：

```
<textarea ng-model="message" cols="40" rows="10"></textarea>
```

两个 **ng-click** 事件，调用控制器函数 **clear()** 和 **save()**：

```
<button ng-click="save()">保存</button>
<button ng-click="clear()">清除</button>
```

一个 **ng-bind** 指令，绑定控制器函数 **left()** 到一个 <span>，字符会向左对齐显示：

```
剩下的字符数：<span ng-bind="left()"></span>
```

两个应用程序库被添加到 HTML 页面：

```
<script src="myTodoApp.js"></script>
<script src="myTodoCtrl.js"></script>
```

AngularJS 参考手册

AngularJS 指令

本教程中使用的 AngularJS 指令：

指令	描述	讲解
ng_app	定义应用程序的根元素。	指令
ng_bind	绑定 HTML 元素到应用程序数据。	简介
ng_click	定义元素被单击时的行为。	HTML 事件
ng_controller	为应用程序定义控制器对象。	控制器
ng_disabled	绑定应用程序数据到 HTML 的 disabled 属性。	HTML DOM
ng_init	为应用程序定义初始值。	指令
ng_model	绑定应用程序数据到 HTML 元素。	指令
ng_repeat	为控制器中的每个数据定义一个模板。	指令
ng_show	显示或隐藏 HTML 元素。	HTML DOM

AngularJS 过滤器

本教程中使用的 AngularJS 过滤器：

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

有关过滤器的具体知识在 AngularJS 过滤器 一章中进行讲解。

W3School jQuery UI教程

来源：jQuery UI教程

整理：飞龙

jQuery UI 基础

jQuery UI 简介

jQuery UI 是一个建立在 jQuery JavaScript 库上的小部件和交互库，您可以使用它创建高度交互的 Web 应用程序。本教程将向您讲解 jQuery UI 是如何工作的。

jQuery UI 特性

简单易用

继承 jQuery 简易使用特性，提供高度抽象接口，短期改善网站易用性。

开源免费

采用 MIT & GPL 双协议授权，轻松满足自由产品至企业产品各种授权需求。

广泛兼容

兼容各主流桌面浏览器。包括IE 6+、Firefox 2+、Safari 3+、Opera 9+、Chrome 1+。

轻便快捷

组件间相对独立，可按需加载，避免浪费带宽拖慢网页打开速度。

标准先进

支持 WAI-ARIA，通过标准 XHTML 代码提供渐进增强，保证低端环境可访问性。

美观多变

提供近 20 种预设主题，并可自定义多达 60 项可配置样式规则，提供 24 种背景纹理选择。

开放公开

从结构规划到代码编写，全程开放，文档、代码、讨论，人人均可参与。

强力支持

Google 为发布代码提供 CDN 内容分发网络支持。

完整汉化

开发包内置包含中文在内的 40 多种语言包。

缺点、不足

- 1.代码不够健壮：缺乏全面的测试用例，部分组件 Bugs 较多，不能达到企业级产品开发要求。
- 2.构架规划不足：组件间 API 缺乏协调，缺乏配合使用帮助。
- 3.控件较少：相对于 Dojo、YUI、Ext JS 等成熟产品，可用控件较少，无法满足复杂界面功能要求。

jQuery UI 下载

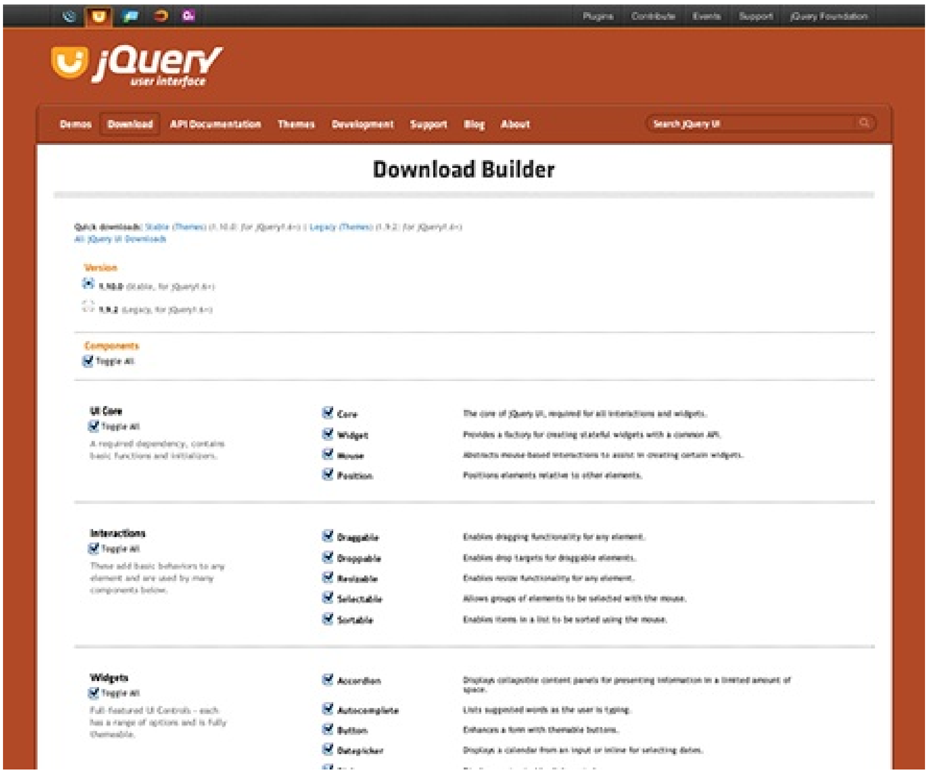
一旦您对 jQuery UI 有了基本了解，您就可以亲自尝试一下。请从 jQuery UI 网站上的 Download Builder（下载生成器） 页面下载 jQuery UI 的副本。

创建自定义 jQuery UI 下载

jQuery UI 的下载生成器（Download Builder）允许您选择您需要下载的组件，为项目获取一个自定义的库版本。创建自定义 jQuery UI 下载需要以下三个步骤：

步骤 1：选择您需要的组件

下载生成器（Download Builder）页面的第一栏列出了 jQuery UI 所有的 JavaScript 组件分类：核心（UI Core）、交互部件（Interactions）、小部件（Widgets）和效果库（Effects）。jQuery UI 中的一些组件依赖于其他组件，当选中这些组件时，它所依赖的其他组件也都会自动被选中。您所选的组件将会合并到一个 jQuery UI JavaScript 文件。



步骤 2：选择一个主题或者自定义一个主题

在下载生成器（Download Builder）页面，您将看到一个文本框，列出了一系列为 jQuery UI 小部件预先设计的主题。您可以从这些提供的主题中选择一个，也可以使用 ThemeRoller 自定义一个主题（详见后面章节的讲解）。

高级主题设置：下载生成器（*Download Builder*）的主题部分也为主题提供了一些高级配置设置。如果您打算在一个页面上使用多个主题，这些字段会派上用场。如果您打算在一个页面上只使用一个主题，那么您完全可以跳过这些设置。

步骤 3：选择 jQuery UI 的版本

在下载生成器（Download Builder）中，最后一步是选择一个版本号。这个步骤很重要，因为 jQuery UI 的版本是配合特定的 jQuery 版本设计的。目前的版本有：

- jQuery UI 1.10.2 – 要求 jQuery 1.6 及以上版本。
- jQuery UI 1.9.2 – 要求 jQuery 1.6 及以上版本。

点击 Download 按钮进行下载！

点击 Download 按钮，完成下载。您将得到一个包含您所选组件的自定义 zip 文件。

jQuery UI 使用

一旦您下载了 jQuery UI，您将得到一个 zip 压缩包，包含下列文件：

- `/css/`
- `/development-bundle/`
- `/js/`
- `index.html`

在网页上使用 jQuery UI

在文本编辑器中打开 `index.html`，您将看到引用了一些外部文件：主题、jQuery 和 jQuery UI。通常情况下，您需要在页面中引用这三个文件，以便使用 jQuery UI 的窗体小部件和交互部件：

```
<link rel="stylesheet" href="css/themename/jquery-ui.custom.css" /
<script src="js/jquery.min.js"></script>
<script src="js/jquery-ui.custom.min.js"></script>
```

一旦您引用了这些必要的文件，您就能向您的页面添加一些 jQuery 小部件。比如，要制作一个日期选择器（datepicker）小部件，您需要向页面添加一个文本输入框，然后再调用 `.datepicker()`，如下所示：

HTML:

```
<input type="text" name="date" id="date" />
```

JavaScript:

```
$( "#date" ).datepicker();
```



如需查看 jQuery UI 小部件和交互部件的实例演示，请访问 jQuery UI 实例。

jQuery UI 定制

jQuery UI 提供了多种定制方式。您已经看到下载生成器（Download Builder）如何定制一个值包含您所需选项的自定义版本，这里还提供了其他定制方式。

jQuery UI 基础：使用选项

jQuery UI 中的每个插件都有一个默认配置，默认配置值一般是根据最基本最常见的使用情况设置的。如果您想要让某个插件设置成非默认值，您可以使用 "options" 重写它的默认设置。选项是一组属性，作为参数传递给 jQuery UI 小部件。例如，滑块（slider）小部件具有 orientation 选项，该选项允许您指定滑块是水平的还是垂直的。为了设置滑块的该选项，您只需将它作为一个参数传递，如下所示：

```
$( "#mySliderDiv" ).slider({
    orientation: "vertical"
});
```

您可以传递更多不同的选项，每个选项之间用逗号分隔：

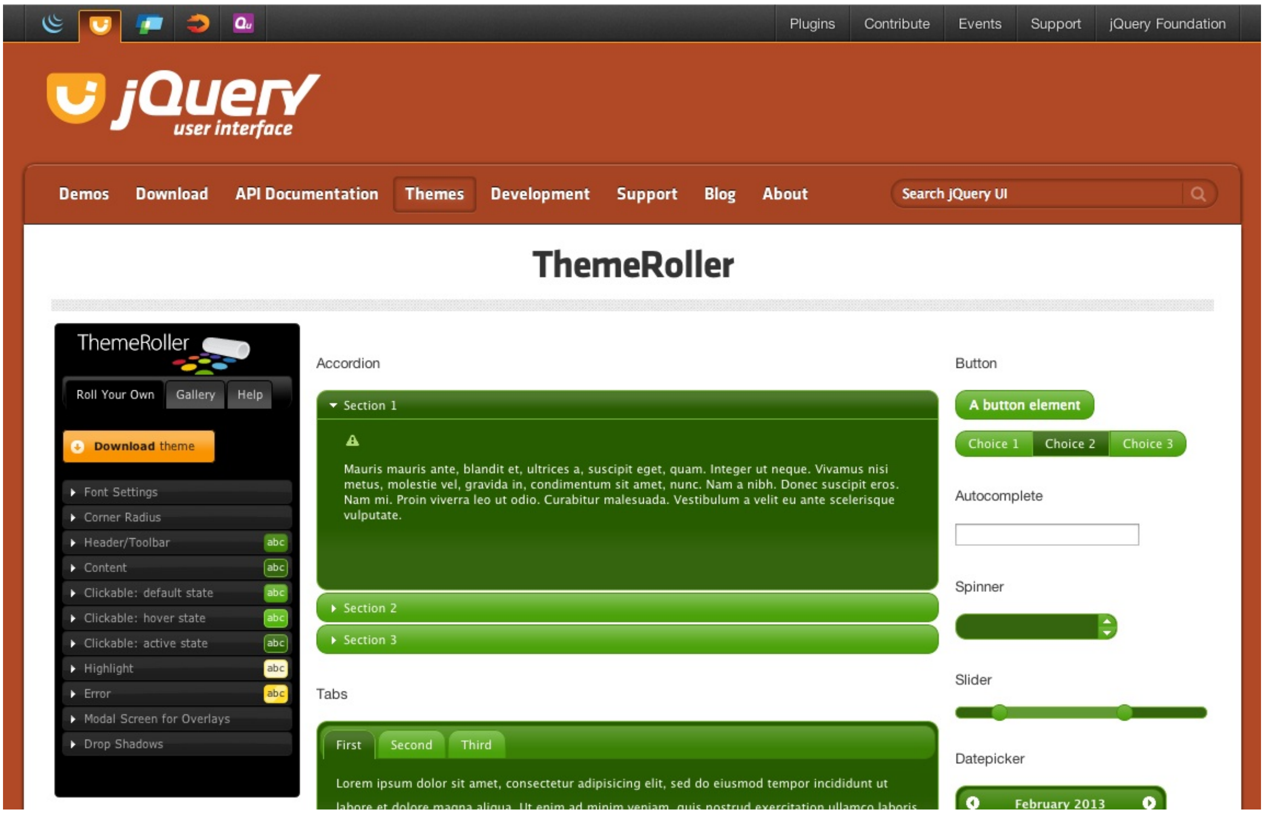
```
$( "#mySliderDiv" ).slider({
    orientation: "vertical",
    min: 0,
    max: 150,
    value: 50
});
```

请记得选项需放在大括号 `{ }` 内。上面的实例只是一个简单的讲解，如需获取整套 jQuery UI 小部件的详细信息，请查看 jQuery UI 实例。

视觉定制：设计一个 jQuery UI 主题

如果您想要设计自己的主题，jQuery UI 提供了一个非常完美的用于主题定制的应用程序，这就是 ThemeRoller。具体定制请访问 jQuery UI ThemeRoller。

ThemeRoller 为所有使用 jQuery UI 小部件设计的元素提供了一个自定义接口。当您调整左栏中的"levers"，右栏中的小部件将根据您的设计进行显示。ThemeRoller 的 Gallery 选项卡提供了一些与设计主题，与下载生成器（Download Builder）页面提供的一样。您可以基于这些主题做调整，或者直接下载。



下载主题

当您点击 ThemeRoller 页面中的 "Download theme" 按钮，将跳转到下载生成器（Download Builder）页面，您的自定义主题会在主体下拉菜单中自动选中。您可以进一步配置下载包。一旦下载完成，您将看到 `example.html` 页面使用了您自定义的主题。

提示： 如果您需要编辑您的主题，只需打开 *CSS* 文件，找到第 *43* 行，*"To view and modify this theme, visit ..."*， 该 *url* 即为在 *ThemeRoller* 中打开主题进行编辑的链接。

jQuery UI 工作原理

jQuery UI 包含了许多维持状态的小部件（Widget），因此，它与典型的 jQuery 插件使用模式略有不同。其安装方式与大部分 jQuery 插件的安装方式类似，jQuery UI 的小部件是基于 部件库（Widget Factory） 创建的，小部件库提供了通用的 API。所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。本教程将通过 进度条（progressbar） 小部件代码实例介绍常见的功能。

安装

为了跟踪部件的状态，我们首先介绍一下小部件的全生命周期。当小部件安装时，生命周期开始。我们只需要在一个或多个元素上调用插件，即安装了小部件。

```
$( "#elem" ).progressbar();
```

这将会初始化 jQuery 对象中的每个元素，在本例中，元素 id 为 "elem"。因为我们调用无参数的 `.progressbar()` 方法，小部件则会按照它的默认选项进行初始化。我们可以在安装时传递一组选项，这样既可重写默认选项。

```
$( "#elem" ).progressbar({ value: 20 });
```

安装时传递的选项数目多少可根据我们的需要而定。任何我们未传递的选项则都使用它们的默认值。

选项是小部件状态的组成部分，所以我们也可以在安装后再进行设置选项。我们将在后续的 `option` 方法中介绍这部分内容。

方法

既然小部件已经初始化，我们就可以查询它的状态，或者在小部件上执行动作。所有初始化后的动作都以方法调用的形式进行。为了在小部件上调用一个方法，我们可以向 jQuery 插件传递方法的名称。例如，为了在进度条（progressbar）小部件上调用 `value` 方法，我们应该使用：

```
$( "#elem" ).progressbar( "value" );
```

如果方法接受参数，我们可以在方法名后传递参数。例如，为了传递参数 `40` 给 `value` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value", 40 );
```

就像 jQuery 中的其他方法一样，大部分的小部件方法为链接返回 jQuery 对象。

```
$( "#elem" )
    .progressbar( "value", 90 )
    .addClass( "almost-done" );
```

公共的方法

每个小部件都有它自己的一套基于小部件所提供功能的方法。然而，有一些方法是所有小部件都共同具有的。

option

正如我们前面所提到的，我们可以在初始化之后通过 `option` 方法改变选项。例如，我们可以通过调用 `option` 方法改变 progressbar（进度条）的 value 为 30。

```
$( "#elem" ).progressbar( "option", "value", 30 );
```

请注意，这与之前我们调用 `value` 方法的实例有所不同。在本实例中，我们调用 `option` 方法，改变 value 选项为 30。

我们也可以为某个选项获取当前的值。

```
$( "#elem" ).progressbar( "option", "value" );
```

另外，我们可以通过给 `option` 方法传递一个对象，一次更新多个选项。

```
$( "#elem" ).progressbar( "option", {
    value: 100,
    disabled: true
});
```

您也许注意到 `option` 方法有着与 jQuery 代码中取值器和设置器相同的标志，就像 `.css()` 和 `.attr()` 。唯一的不同就是您必须传递字符串 "option" 作为第一个参数。

disable

`disable` 方法禁用小部件。在进度条（progressbar）实例中，这会改变样式让进度条显示为禁用状态。

```
$( "#elem" ).progressbar( "disable" );
```

调用 `disable` 方法等同于设置 `disabled` 选项为 `true` 。

enable

`enable` 方法是 `disable` 方法的对立面。

```
$( "#elem" ).progressbar( "enable" );
```

调用 `enable` 方法等同于设置 `disabled` 选项为 `false` 。

destroy

如果您不再需要小部件，那么可以销毁它，返回到最初的标记。这意味着小部件生命周期的终止。

```
$( "#elem" ).progressbar( "destroy" );
```

一旦您销毁了一个小部件，您就不能在该部件上调用任何方法，除非您再次初始化这个小部件。如果您要移除元素，可以直接通过 `.remove()` ，也可以通过 `.html()` 或 `.empty()` 修改祖先，小部件会自动销毁。

widget

一些小部件生成包装器元素，或与原始元素断开连接的元素。在下面的实例中， `widget` 将返回生成的元素。在进度条（progressbar）实例中，没有生成的包装器， `widget` 方法返回原始的元素。

```
$( "#elem" ).progressbar( "widget" );
```

事件

所有的小部件都有跟他们各种行为相关的事件，用于在状态改变时通知您。对于大多数的小部件，当事件被触发时，名称以小部件名称为前缀。例如，我们可以绑定进度条（）的 change 事件，一旦值发生变化时就触发。

```
$( "#elem" ).bind( "progressbarchange", function() {
    alert( "The value has changed!" );
});
```

每个事件都有一个相对应的回调，作为选项进行呈现。我们可以使用进度条（progressbar）的 `change` 回调，这等同于绑定 `progressbarchange` 事件。

```
$( "#elem" ).progressbar({
    change: function() {
        alert( "The value has changed!" );
    }
});
```

公共的事件

大多数事件是针对特定的小部件，所有的小部件都有一个公共的 `create` 事件。该事件在小部件被创建时即被触发。

jQuery UI 主题

jQuery UI 主题

所有的 jQuery UI 插件都允许开发人员无缝集成 UI 小部件到他们网站或应用程序的外观和感观。每个插件通过 CSS 定义样式，且包含了两层样式信息：标准的 jQuery UI CSS 框架 样式和具体的插件样式。

jQuery UI CSS 框架提供了语义表示的类，用来表明小部件内元素的角色，比如标题、内容或可点击区域。这些在所有的小部件中都是一致的，一个可点击的 tab（标签页）、accordion（折叠面板）或 button（按钮）都有相同的 `ui-state-default` class，用来表明它们是可点击的。当用户鼠标悬浮在这些元素上面时，这个 class 就变成 `ui-state-hover` ，当选中这些元素时则变成 `ui-state-active` 。这些 class 的一致性使得所有部件中具有相似角色或交互状态的元素在外观表现上一致。

CSS 框架样式封装在一个单独的文件中，名为 `ui.theme.css` 。这个文件时通过 ThemeRoller 应用程序来修改的。框架样式只包含影响外观和感观的属性，只要是颜色、背景图像、图标等。所以这些是 "安全的" 样式，不会影响到插件的功能。这种分隔意味着开发人员可以通过在 `theme.css` 文件中修改颜色和图像来创建一个自定义的外观和感观。由于未来的插件或者 bug 修复将是可用的，这些不通过修改即可与主题一起使用。

由于框架样式只覆盖了外观和感观，所以还需要包含具体的插件样式表，这些样式表包括了所有额外的让小部件具有功能性的结构样式规则，比如尺寸、内边距、外边距、定位、浮动。每个插件的样式表位于 `themes/base` 文件夹内，且配合插件进行命名，比如 "jquery.ui.accordion.css"。这些样式必须认真编辑，因为它们与脚本一起提供了框架样式的覆盖。

我们鼓励所有的开发人员创建 jQuery 插件，jQuery UI CSS 框架使得最终用户更容易定制主题和使用插件。

主题化

下面列出了三种主题化 jQuery UI 插件的一般方法：

- 下载 **ThemeRoller** 主题：最早的创建主题的方式是使用 ThemeRoller 来生成和下载一个主题。这个应用程序将创建一个新的 `ui.theme.css` 文件和一个包含了所有必需的背景图像及图标精灵的 `images` 文件夹。这个方法是最早的创建和维护主题的方式，但是它对 ThemeRoller 中提供的选项限制了自定义。
- 修改 **CSS** 文件：为了对外观和感观做进一步的控制，您可以选择从默认主题（Smoothness）开始，或者从一个由 ThemeRoller 生成的主题开始，然后调整 `ui.theme.css` 文件，或者任意一个独立插件的样式表。例如，您可以很容易地调整所有按钮的角半径为不同于其他 UI 组件的值，或者使用自定义设置为图标精灵改变路径。通过一点点的样式范围，您甚至可以在一个 UI 中同时使用多个主题。为了易于维护，建议只更改 `ui.theme.css` 文件和图像。
- 重新编写自定义的 **CSS**：为了最大程度地控制外观和感观，可以重新开始编写

每个插件的 CSS，而不使用框架类或者特定的插件样式表。如果想要的外观和感观不能通过修改 CSS 来实现或者使用高度自定义的标记，那么就可以采用这个方法。这个方法要求在 CSS 方面有深厚的专业知识，且要求手动更新未来的插件。

使用 **ThemeRoller**、**jQuery UI CSS** 框架，以及设计自定义主题

- jQuery UI ThemeRoller
- jQuery UI CSS 框架 API
- 设计主题

jQuery UI ThemeRoller

ThemeRoller 简介

ThemeRoller 是一个 Web 应用程序，为 jQuery UI 设计和下载自定义主题提供了直观的界面。您可以访问 jQuery UI ThemeRoller 进行主题定制。

jQuery UI ThemeRoller 是由波士顿的 Filament Group, Inc 设计和开发的。

ThemeRoller 界面

ThemeRoller 的界面分为不同面板，各面板分别是全局字体和圆角半径设置、小部件容器样式、可点击元素的互动状态，及覆盖和阴影的各种样式。这些面板允许配置各种 CSS 属性，比如字体尺寸、颜色、粗细、背景颜色和纹理、边框颜色、文本颜色、图标颜色、圆角半径，等等。

主题馆（Gallery）：预先设计主题

ThemeRoller 主题可以通过永久链接 URL 进行查看，它包含一些预先设计的主题可供选择。主题馆（Gallery）可以通过位于应用程序界面顶端的标签栏进行访问。从主题馆（Gallery）中，您可以预览和下载主题，甚至可以选择一个主题，然后切换到 "Roll Your Own" 标签页进行调整。

下载主题

当您设计完主题后，您可以下载主题以便在项目中进行使用。ThemeRoller 在顶部有一个 "Download theme（下载主题）" 按钮，可以生成一个 zip 压缩包，包含了所有的主题相关文件。下载文件中的图像是按照您的规格进行生成的，并保存为高质量的 PNG 文件。

您的主题将包含图像和 CSS，组成了 jQuery UI CSS 框架的自定义版本，包含了所有插件的图像和 CSS。

在项目中按照下载的主题

一旦下载了主题，解压缩后，您将看到一个名为 `themes` 的文件夹。该文件夹包括了主题所需的 CSS 和图像。复制主题文件夹到您的项目中，并在页面中链接 `themes/all.css` 文件。

创建自定义的 "ThemeRoller-Ready" 组件





ThemeRoller 生成了一个 jQuery UI CSS 框架的自定义版本，用于开发您自己的 ThemeRoller-ready jQuery 组件。通过这个框架生成的类被设计来提供通用的用户界面的设计情况，包括状态、图标以及各种辅助类。

如需了解更多 jQuery UI CSS 框架的开发信息，请查看 主题化 API 文档。

ThemeRoller 链接

- 教程：开发您自己的 jQuery "ThemeRoller-Ready" 组件，Filament Group
- ThemeRoller 简介：设计 & 下载 jQuery UI 自定义主题，Filament Group
- 视频：设计 & 下载 jQuery UI 自定义主题，Filament Group

jQuery UI CSS 框架 API

jQuery UI CSS 框架

jQuery UI 包含了一个强大的 CSS 框架，为了创建自定义 jQuery 小部件而设计的。框架包含了通用的用户界面所需的类，且可使用 jQuery UI ThemeRoller 进行维护。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件。您需采用共享标记公约，以便在插件社区的代码集成。

框架类

下面的 CSS 类根据样式是否是固定的结构化的，或者是否是可主题化的（颜色、字体、背景等），分别定义在 `ui.core.css` 和 `ui.theme.css` 两个文件中。这些类被设计来用于用户界面元素，以便获得整个应用程序的视觉一致性，可通过 jQuery UI ThemeRoller 对组件进行主题化。

布局助手

- `.ui-helper-hidden` ：对元素应用 `display: none` 。
- `.ui-helper-hidden-accessible` ：对元素应用访问隐藏（通过页面绝对定位）。
- `.ui-helper-reset` ：UI 元素的基本样式重置。重置的元素比如： `padding` 、 `margin` 、 `text-decoration` 、list-style，等等。
- `.ui-helper-clearfix` ：对父元素应用浮动包装属性。
- `.ui-helper-zfix` ：对 `<iframe>` 元素应用 iframe "fix" CSS。

小部件容器

- `.ui-widget` ：对所有小部件的外部容器应用的 Class。对小部件应用字体和字体尺寸，同时也对自表单元素应用相同的字体和 1em 的字体尺寸，以应对 Windows 浏览器中的继承问题。
- `.ui-widget-header` ：对标题容器应用的 Class。对元素及其子元素的文本、链接、图标应用标题容器样式。
- `.ui-widget-content` ：对内容容器应用的 Class。对元素及其子元素的文本、链接、图标应用内容容器样式。（可应用到标题的父元素或者同级元素）

交互状态

- `.ui-state-default` ：对可点击按钮元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable default" 容器样式。
- `.ui-state-hover` ：当鼠标悬浮在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-focus` ：当键盘聚焦在可点击按钮元素上时应用的 Class。对元

素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-active` ：当鼠标点击可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable active" 容器样式。

交互提示 Cues

- `.ui-state-highlight` ：对高亮或者选中元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "highlight" 容器样式。
- `.ui-state-error` ：对错误消息容器元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "error" 容器样式。
- `.ui-state-error-text` ：对只有无背景的错误文本颜色应用的 Class。可用于表单标签，也可以对子图标应用错误图标颜色。
- `.ui-state-disabled` ：对禁用的 UI 元素应用一个暗淡的不透明度。意味着对一个已经定义样式的元素添加额外的样式。
- `.ui-priority-primary` ：对第一优先权的按钮应用的 Class。应用粗体文本。
- `.ui-priority-secondary` ：对第二优先权的按钮应用的 Class。应用正常粗细的文本，对元素应用轻微的透明度。

图标

状态和图像

- `.ui-icon` ：对图标元素应用的基本 Class。设置尺寸为 16px 方块，隐藏内部文本，对 "content" 状态的精灵图像设置背景图像。注意： *`.ui-icon` class* 将根据它的父容器得到一个不同的精灵背景图像。例如， *`ui-state-default`* 容器内的 *`ui-icon`* 元素将根据 *`ui-state-default`* 的图标颜色进行着色。

图标类型

在声明 `.ui-icon` class 之后，接着您可以声明一个秒速图标类型的 class。通常情况下，图标 class 遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}` 。

例如，一个指向右侧的三角形图标，如下所示： `.ui-icon-triangle-1-e`

jQuery UI 的 ThemeRoller 在它的预览一栏中提供了全套的 CSS 框架图标。将鼠标悬浮在图标上可查看 class 名称。

其他视觉效果

圆角半径助手

- `.ui-corner-tl` ：对元素的左上角应用圆角半径。

- `.ui-corner-tr` ：对元素的右上角应用圆角半径。
- `.ui-corner-bl` ：对元素的左下角应用圆角半径。
- `.ui-corner-br` ：对元素的右下角应用圆角半径。
- `.ui-corner-top` ：对元素上边的左右角应用圆角半径。
- `.ui-corner-bottom` ：对元素下边的左右角应用圆角半径。
- `.ui-corner-right` ：对元素右边的上下角应用圆角半径。
- `.ui-corner-left` ：对元素左边的上下角应用圆角半径。
- `.ui-corner-all` ：对元素的所有四个角应用圆角半径。

覆盖 & 阴影

- `.ui-widget-overlay` ：对覆盖屏幕应用 100% 宽度和高度，同时设置背景颜色/纹理和屏幕不透明度。
- `.ui-widget-shadow` ：对覆盖应用的 Class，设置了不透明度、上偏移/左偏移，以及阴影的 "厚度"。厚度是通过对阴影所有边设置内边距（padding）进行应用的。偏移是通过设置上外边距（margin）和左外边距（margin）进行应用的（可以是正数，也可以是负数）。

jQuery UI 设计主题

文件结构

主题是以特定的方式来增加他们的易用性。通常，文件目录结构如下所示：

- `themename/` – 您的主题必须完全包含在一个单独的以主题名称命名的文件夹内。
- `themename/themename.css` – 这是基本的 CSS 文件。无论使用了哪个插件，该文件都必须在每个使用主题的页面中引用。该文件应该是轻量级的，只包括要点。
- `themename/themename.pluginname.css` – 您支持的每个插件都需要一个 CSS 文件。插件的名称应直接包含在文件名中。例如，如果您为 tabs（标签页）插件进行主题化，则有： `themename.tabs.js` 。
- `themename/img.png` – 您的主题可以包含图像。它们可以根据您的喜好进行命名，这里没有特定的命名惯例。

如需了解主题文件结构是如何完成的实例，请访问 jQuery UI 基本主题。

定义样式

为主题编写样式是非常简单的，这是因为主题的灵活性。

所有的主题都应该有一个基本的 CSS class。这个主要的 class 允许用户启用禁用主题。您的根 class 的格式应该是 `.ui-themename` 。且它在 HTML 文件中的用法如下所示：

```
<html>
<head>
    <title>My Site</title>
    <link rel="stylesheet" href="themename/themename.css" />
    <link rel="stylesheet" href="othertheme/othertheme.css" />
    <link rel="stylesheet" href="othertheme/othertheme.dialog.css"
</head>
<body class="ui-themename">
    <div class="ui-othertheme">
        <div class="ui-dialog">This is a modal dialog.</div>
    </div>
</body>
</html>
```

在上面的实例中，发生了一些重要的事情：

- 我们同时向文档中加载两个主题。
- 整个页面机器所有内容，是根据 themename 的样式进行主题化的。

- 然而，带有 ui-othertheme class 的 `<div>` 及其中的每个元素（包括模态对话框）都是根据 othertheme 的样式进行主题化的。

如果我们打开 `themename.css` 文件进行查看，我们可以看到如下代码：

```
body.ui-themename { background:#111; color:snow; }
.ui-themename a, a.ui-themename { color:#68D; outline:none; }
.ui-themename a:visited, a.ui-themename:visited { color:#D66; }
.ui-themename a:hover, a.ui-themename:hover { color:#FFF; }
```

请注意， `themename.css` 文件只包括全局通用的样式信息，特定插件的样式信息不在这里进行定义。这里的样式对所有主题都是适用的。不用担心一个主题会占据多个文件 - 这些会在创建和下载的过程被简化。

jQuery UI 部件库

jQuery UI 部件库（Widget Factory）

jQuery UI 部件库（Widget Factory）是一个可扩展的基础，所有的 jQuery UI 小部件都是在上面进行创建的。使用部件库（Widget Factory）来创建插件，提供了方便的状态管理，同时也为一些常见的任务提供了便捷，比如暴露插件方法，实例化后改变选项等。

jQuery UI 通过部件库（Widget Factory）扩展小部件

jQuery UI 的部件库（Widget Factory）使得创建小部件变得更加容易，这些小部件扩展了已有小部件的功能。这样子您就能在已有的基础上创建出功能强大的小部件，也可以在已有的小部件功能上做细微的调整。

注意：在学习本章节之前，需要明白什么是部件库（Widget Factory），及它是怎么工作的。如果您对这些知识还不熟悉，那么请先查看如何使用部件库（Widget Factory）章节。

创建小部件扩展

通过部件库（Widget Factory）创建小部件是通过向 `$.widget()` 传递小部件名称和一个原型对象来完成的。下面的实例是在 "custom" 命名空间中创建一个 "superDialog" 小部件。

```
$.widget( "custom.superDialog", {} );
```

为了支持扩展，`$.widget()` 可选性地接受作为父部件使用的小部件的构造函数。当指定一个父部件时，把它作为第二个参数进行传递，放在小部件名称后面，在小部件原型对象前面。

就像上面的实例，下面也要在 "custom" 命名空间中创建一个 "superDialog" 小部件。但是这次传递的是 jQuery UI 的 dialog（对话框）小部件 的构造函数（`$.ui.dialog`），表示 superDialog 小部件应该使用 jQuery UI 的 dialog（对话框）小部件作为父部件。

```
$.widget( "custom.superDialog", $.ui.dialog, {} );
```

在这里，superDialog 和 dialog 两个小部件实质上是等价的，只是名称和命名空间不同而已。为了让我们新的小部件更具特点，我们可以添加一些方法到它的原型对象上。

小部件的原型对象是传递给 `$.widget()` 的最后一个参数。到目前为止，我们的实例使用的是一个空的对象。现在让我们给这个对象添加一个方法：

```
$.widget( "custom.superDialog", $.ui.dialog, {
    red: function() {
        this.element.css( "color", "red" );
    }
});

// Create a new <div>, convert it into a superDialog, and call the
$( "<div>I am red</div>" )
    .superDialog()
    .superDialog( "red" );
```

现在 `superDialog` 有一个 `red()` 方法，这会把它的文本颜色改为红色。请注意，部件库（Widget Factory）是如何自动设置 `this` 为小部件的实例对象。如需了解实例上所有可用的方法和属性列表，请访问 部件库（Widget Factory） API 文档。

扩展已有的方法

有时候，您需要调整或添加已有部件方法的行为。您可以把方法名称指定为原型对象上需要重载的方法名称。下面的实例重载了 dialog（对话框）的 `open()` 方法。由于对话框默认是打开的，当运行这段代码时， `"open"` 将会被记录。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
    }
});

// Create a new <div>, and convert it into a superDialog.
$( "<div>" ).superDialog();
```

当运行这段代码时，有一个问题。由于我们重载了 `open()` 的默认行为，所以 dialog（对话框）不再显示在屏幕上。

当我们在原型对象上使用方法，我们实际上是重载了原始的方法，在原型链中使用了一个新的方法。

为了让父部件方法可用，部件库（Widget Factory）提供了两个方法 - `_super()` 和 `_superApply()` 。

使用 `_super()` 和 `_superApply()` 来访问父部件

`_super()` 和 `_superApply()` 在父部件中调用了同样的方法。请看下面的实例。就像上一个实例，这个实例也重载了 `open()` 方法来记录 `"open"` 。然而，这次运行 `_super()` 是调用了 dialog（对话框）的 `open()` ，并打开对话框。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );

        // Invoke the parent widget's open().
        return this._super();
    }
});

$( "<div>" ).superDialog();
```

`_super()` 和 `_superApply()` 实际上等同于最初的 `Function.prototype.call()` 和 `Function.prototype.apply()` 方法。因此，`_super()` 接受一个参数列表，`_superApply()` 接受一个数组作为参数。下面的实例演示了这二者之间的不同。

```
$.widget( "custom.superDialog", $.ui.dialog, {
    _setOption: function( key, value ) {

        // Both invoke dialog's setOption() method. _super() requi
        // be passed as an argument list, _superApply() as a single
        this._super( key, value );
        this._superApply( arguments );
    }
});
```

重定义小部件

jQuery UI 1.9 添加了重定义小部件的功能。因此，可以不用创建一个新的小部件，我们只需要传递 `$.widget()` 这样一个已有的小部件名称和构造函数即可。下面的实例在 `open()` 中添加了相同的记录，但不是通过创建一个新的小部件来完成的。

```
$.widget( "ui.dialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
        return this._super();
    }
});

$( "<div>" ).dialog();
```

通过这个方法，我们可以扩展一个已有的小部件方法，但是仍然可以使用 `_super()` 来访问原始的方法 - 这些都不是通过创建一个新的小部件来完成的，而是直接重定义小部件即可。

小部件（Widgets）和多态性（Polymorphism）

当在小部件扩展及它们的插件之间进行交互时候，有一点值得注意，父部件的插件不能用来调用子部件元素上的方法。下面的实例演示了这一点。

```
$.widget( "custom.superDialog", $.ui.dialog, {} );

var dialog = $( "<div>" ).superDialog();

// This works.
dialog.superDialog( "close" );

// This doesn't.
dialog.dialog( "close" );
```

上面的实例中，父部件的插件， `dialog()` ，不能调用 superDialog 元素上的 `close()` 方法。如需了解更多调用小部件方法的知识，请查看 小部件（Widget）方法调用。

定制个性化实例

目前为止，我们看到的实例都有在小部件原型上扩展的方法。在原型上重载的方法影响了小部件的所有实例。

为了演示这一点，请看下面的实例。dialog（对话框）的两个势力都使用了相同的 `open()` 方法。

```
$.widget( "ui.dialog", $.ui.dialog, {
    open: function() {
        console.log( "open" );
        return this._super();
    }
});

// Create two dialogs, both use the same open(), therefore "open" i
$( "<div>" ).dialog();
$( "<div>" ).dialog();
```

有时候，您只需要改变小部件的某个实例的行为。为了做到这点，您需要使用正常的 JavaScript 属性分配，获得对实例的引用，并重载该方法。具体如下面实例所示。

```
var dialogInstance = $( "<div>" )
    .dialog()
    // Retrieve the dialog's instance and store it.
    .data( "ui-dialog" );

// Override the close() method for this dialog
dialogInstance.close = function() {
    console.log( "close" );
};

// Create a second dialog
$( "<div>" ).dialog();

// Select both dialogs and call close() on each of them.
// "close" will only be logged once.
$( ":data(ui-dialog)" ).dialog( "close" );
```

个性化实例的重载方法技术是完美的一次性定制。

jQuery UI 小部件（Widget）方法调用

小部件（Widget）是通过 部件库（Widget Factory） 使用方法来改变他们初始化后的状态和执行动作而被创建的。有两种调用小部件方法的方式 - 通过部件库（Widget Factory）创建的插件，或者通过调用元素实例对象上的方法。

插件调用

使用小部件的插件调用方法，把方法名称以字符串形式进行传递。例如，点击这里查看，如何调用 dialog（对话框）小部件的 `close()` 方法。

```
$( ".selector" ).dialog( "close" );
```

如果方法要求参数，请作为额外的参数传递给插件。点击这里查看，如何调用 dialog（对话框）的 `option()` 方法。

```
$( ".selector" ).dialog( "option", "height" );
```

这会返回 dialog（对话框）的 `height` 选项 的值。

实例调用

每个小部件的每个实例都是使用 `jQuery.data()` 存储在元素上。为了检索实例对象，请使用小部件的全称作为键名调用 `jQuery.data()` 。具体如下面实例所示。

```
var dialog = $( ".selector" ).data( "ui-dialog" );
```

在您引用实例对象之后，可以直接在上面调用方法。

```
var dialog = $( ".selector" ).data( "ui-dialog" );
dialog.close();
```

在 jQuery UI 1.11 中，新的 `instance()` 方法会使得这个过程变得更简单。

```
$( ".selector" ).dialog( "instance" ).close();
```

返回类型

大多数通过小部件的插件调用的方法将返回一个 `jQuery` 对象，所以方法调用可以通过额外的 jQuery 方法链接。当在实例上进行调用时，则会返回 `undefined` 。具体如下面实例所示。

```
var dialog = $( ".selector" ).dialog();

// Instance invocation - returns undefined
dialog.data( "ui-dialog" ).close();

// Plugin invocation - returns a jQuery object
dialog.dialog( "close" );

// Therefore, plugin method invocation makes it possible to
// chain method calls with other jQuery functions
dialog.dialog( "close" )
    .css( "color", "red" );
```

例外的是，返回小部件相关信息的那些方法。例如 dialog（对话框）的 `isOpen()` 方法。

```
$( ".selector" )
    .dialog( "isOpen" )
    // This will throw a TypeError
    .css( "color", "red" );
```

这会产生一个 `TypeError` 错误，因为 `isOpen()` 返回的是一个布尔值，而不是一个 jQuery 对象。

jQuery UI 为什么使用部件库（Widget Factory）

编写 jQuery 插件与向 `jQuery.prototype` （通常显示为 `$.fn` ）添加方法一样简单，且需要遵循一些简单的规则，比如返回 `this` 。所以为什么会存在部件库（Widget Factory）？

在本章节中，我们将讲解部件库（Widget Factory）的好处，并了解何时使用它，以及为什么要使用它。

无状态 vs. 有状态插件

大多数 jQuery 插件是无状态的，它们执行一些动作即完成了它们的任务。例如，如果您使用 `.text( "hello" )` 设置元素的文本，没有安装阶段，结果都是一样的。对于这种类型的插件，它只是扩展了 jQuery 的原型。

然而，一些插件是有状态的，它们有全生命周期、维持状态以及对变化的反应。这些插件需要大量专门的代码来初始化和状态管理（有时是销毁）。这就导致出现了用于创建有状态插件的模板。更糟糕的是，每个插件的作者按照不同的方式进行管理插件的生命周期和状态，这就导致了不同的插件有不同的 API 样式。部件库（Widget Factory）旨在解决这些问题，它移除了模板，并为插件创建了一个一致的 API。

一致的 API

部件库（Widget Factory）定义了如何创建和销毁小部件，获取和设置选项，调用方法，以及监听小部件触发的事件。通过使用部件库（Widget Factory）来创建有状态的插件，会自动符合定义的标准，让新用户更容易使用您的插件。另外，部件库（Widget Factory）还能实现定义接口的功能。如果您对部件库（Widget Factory）提供的 API 还不熟悉，请查看 如何使用部件库（Widget Factory）。

在初始化时设置选项

当您创建一个接受选项的插件时，您应该为尽可能多的选项定义 defaults。然后在初始化时，把用户提供的选项与 defaults 进行合并。您也可以暴露 defaults，这样用户就可以更改默认值。在 jQuery 插件中，一个常用的模式如下所示：

```
$.fn.plugin = function( options ) {
    options = $.extend( {}, $.fn.plugin.defaults, options );
    // Plugin logic goes here.
};

$.fn.plugin.defaults = {
    param1: "foo",
    param2: "bar",
    param3: "baz"
};
```

部件库（Widget Factory）也提供了这个功能，并在这上面做了改进。使用部件库（Widget Factory）之后，它将如下所示。

```
$.widget( "ns.plugin", {

    // Default options.
    options: {
        param1: "foo",
        param2: "bar",
        param3: "baz"
    },

    _create: function() {
        // Options are already merged and stored in this.options
        // Plugin logic goes here.
    }

});
```

jQuery UI 如何使用部件库（Widget Factory）

我们将创建一个进度条。正如下面实例所示，这可以通过调用 `jQuery.widget()` 来完成，它带有两个参数：一个是要创建的插件名称，一个是包含支持插件的函数的对象文字。当插件被调用时，它将创建一个新的插件实例，所有的函数都将在该实例的语境中被执行。这与两种重要方式的标准 jQuery 插件不同。首先，语境是一个对象，不是 DOM 元素。其次，语境总是一个单一的对象，不是一个集合。

```
$.widget( "custom.progressbar", {
    _create: function() {
        var progress = this.options.value + "%";
        this.element
            .addClass( "progressbar" )
            .text( progress );
    }
});
```

插件的名称必须包含命名空间，在这个实例中，我们使用了 `custom` 命名空间。您只能创建一层深的命名空间，因此， `custom.progressbar` 是一个有效的插件名称，而 `very.custom.progressbar` 不是一个有效的插件名称。

我们看到部件库（Widget Factory）为我们提供了两个属性。 `this.element` 是一个包含一个元素的 jQuery 对象。如果我们的插件在包含多个元素的 jQuery 对象上调用，则会为每个元素创建一个单独的插件实例，且每个实例都会有自己的 `this.element` 。第二个属性， `this.options` ，是一个包含所有插件选项的键名/键值对的哈希（hash）。这些选项可以被传给插件，如下所示：

```
$( "<div></div>" )
    .appendTo( "body" )
    .progressbar({ value: 20 });
```

当我们调用 `jQuery.widget()` ，它通过给 `jQuery.fn` （用于创建标准插件的系统） 添加函数来扩展 jQuery。所添加的函数名称是基于您传给 `jQuery.widget()` 的名称，不带命名空间 - "progressbar"。传给插件的选项是在插件实例中获取设置的值。如下面的实例所示，我们可以为任意一个选项指定默认值。当设计您的 API 时，您应该清楚你的插件的最常见的使用情况，以便您可以设置适当的默认值，且确保使所有的选项真正可选。

```
$.widget( "custom.progressbar", {

    // Default options.
    options: {
        value: 0
    },
    _create: function() {
        var progress = this.options.value + "%";
        this.element
            .addClass( "progressbar" )
            .text( progress );
    }
});
```

调用插件方法

现在我们可以初始化我们的进度条，我们将通过在插件实例上调用方法来执行动作。为了定义一个插件方法，我们只在我们传给 `jQuery.widget()` 的对象中引用函数。我们也可以通过给函数名加下划线前缀来定义 "private" 方法。

```
$.widget( "custom.progressbar", {

    options: {
        value: 0
    },

    _create: function() {
        var progress = this.options.value + "%";
        this.element
            .addClass( "progressbar" )
            .text( progress );
    },

    // Create a public method.
    value: function( value ) {

        // No value passed, act as a getter.
        if ( value === undefined ) {
            return this.options.value;
        }

        // Value passed, act as a setter.
        this.options.value = this._constrain( value );
        var progress = this.options.value + "%";
        this.element.text( progress );
    },

    // Create a private method.
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    }
});
```

为了在插件实例上调用方法，您可以向 jQuery 插件传递方法的名称。如果您调用的方法接受参数，您只需简单地在方法名后面传递这些参数即可。

注意：通过向同一个用于初始化插件的 jQuery 函数传递方法名来执行方法。这样做是为了在保持链方法调用时防止 jQuery 命名空间污染。在本章节的后续，我们将看到看起来更自然的其他用法。

```
var bar = $( "<div></div>" )
    .appendTo( "body" )
    .progressbar({ value: 20 });

// Get the current value.
alert( bar.progressbar( "value" ) );

// Update the value.
bar.progressbar( "value", 50 );

// Get the current value again.
alert( bar.progressbar( "value" ) );
```

使用选项

`option()` 方法是自动提供给插件的。 `option()` 方法允许您在初始化后获取并设置选项。该方法像 jQuery 的 `.css()` 和 `.attr()` 方法：您可以只传递一个名称作为取值器来使用，也可以传递一个名称和值作为设置器使用，或者传递一个键名/键值对的哈希来设置多个值。当作为取值器使用时，插件将返回与传入名称相对应的选项的当前值。当作为设置器使用时，插件的 `_setOption` 方法将被每个被设置的选项调用。我们可以在我们的插件中指定一个 `_setOption` 方法来反应选项更改。对于更改选项要独立执行的动作，我们可以重载 `_setOptions` 。

```
$.widget( "custom.progressbar", {
    options: {
        value: 0
    },
    _create: function() {
        this.options.value = this._constrain(this.options.value);
        this.element.addClass( "progressbar" );
        this.refresh();
    },
    _setOption: function( key, value ) {
        if ( key === "value" ) {
            value = this._constrain( value );
        }
        this._super( key, value );
    },
    _setOptions: function( options ) {
        this._super( options );
        this.refresh();
    },
    refresh: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
    },
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    }
});
```

添加回调

最简单的扩展插件的方法是添加回调，这样用户就可以在插件状态发生变化时做出反应。我们可以看下面的实例如何在进度达到 100% 时添加回调到进度条。 `_trigger()` 方法有三个参数：回调名称，一个启动回调的 jQuery 事件对象，以及一个与事件相关的数据哈希。回调名称是唯一一个必需的参数，但是对于想要在插件上实现自定义功能的用户，其他的参数是非常有用的。例如，如果我们创建一个可拖拽插件，我们可以在触发拖拽回调时传递 mousemove 事件，这将允许用户对基于由事件对象提供的 x/y 坐标上的拖拽做出反应。请注意，传递到 `_trigger()` 的原始的事件必须是一个 jQuery 事件，而不是一个原生的浏览器事件。

```
$.widget( "custom.progressbar", {
    options: {
        value: 0
    },
    _create: function() {
        this.options.value = this._constrain(this.options.value);
        this.element.addClass( "progressbar" );
        this.refresh();
    },
    _setOption: function( key, value ) {
        if ( key === "value" ) {
            value = this._constrain( value );
        }
        this._super( key, value );
    },
    _setOptions: function( options ) {
        this._super( options );
        this.refresh();
    },
    refresh: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
        if ( this.options.value == 100 ) {
            this._trigger( "complete", null, { value: 100 } );
        }
    },
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    }
});
```

回调函数本质上只是附加选项，所以您可以像其他选项一样获取并设置它们。无论何时执行回调，都会有一个相对应的事件被触发。事件类型是通过连接插件的名称和回调函数名称确定的。回调和事件都接受两个相同的参数：一个事件对象和一个与事件相关的数据哈希，具体如下面实例所示。您的插件可能需要包含防止用户使用的功能，为了做到这点，最好的方法就是创建爱你可撤销的回调。用户可以撤销回调或者相关的事件，与他们撤销任何一个原生事件一样，都是通过调用 `event.preventDefault()` 或返回 `false` 来实现的。如果用户撤销回调，`_trigger()` 方法将返回 `false`，这样您就能在插件内实现合适的功能。

```
var bar = $( "<div></div>" )
    .appendTo( "body" )
    .progressbar({
        complete: function( event, data ) {
            alert( "Callbacks are great!" );
        }
    })
    .bind( "progressbarcomplete", function( event, data ) {
        alert( "Events bubble and support many handlers for extreme
        alert( "The progress bar value is " + data.value );
    });

bar.progressbar( "option", "value", 100 );
```

本质

现在我们已经看到如何使用部件库（Widget Factory）创建一个插件，接下来让我们看看它实际上是如何工作的。当您调用 `jQuery.widget()` 时，它将为插件创建一个构造函数，并设置您为插件实例传入的作为原型的对象。所有自动添加到插件的功能都来自一个基本的小部件原型，该原型定义为 `jQuery.Widget.prototype` 。当创建插件实例时，会使用 `jQuery.data` 把它存储在原始的 DOM 元素上，插件名作为键名。

由于插件实例直接链接到 DOM 元素上，您可以直接访问插件实例，而不需要遍历插件方法。这将允许您直接在插件实例上调用方法，而不需要传递字符串形式的方法名，同时您也可以直接访问插件的属性。

```
var bar = $( "<div></div>" )
    .appendTo( "body" )
    .progressbar()
    .data( "progressbar" );

// Call a method directly on the plugin instance.
bar.option( "value", 50 );

// Access properties on the plugin instance.
alert( bar.options.value );
```

您也可以在不遍历插件方法的情况下创建一个实例，通过选项和元素直接调用构造函数即可：

```
var bar = $.custom.progressbar( {}, $( "<div></div>" ).appendTo( "b

// Same result as before.
alert( bar.options.value );
```

扩展插件的原型

插件有构造函数和原型的最大好处是易于扩展插件。通过添加或修改插件原型上的方法，我们可以修改插件所有实例的行为。例如，如果我们想要向进度条添加一个方法来重置进度为 0%，我们可以向原型添加这个方法，它将在所有插件实例上可调用。

```
$.custom.progressbar.prototype.reset = function() {
    this._setOption( "value", 0 );
};
```

如需了解扩展小部件的更多细节，以及如何在一个已有的小部件上创建一个全新的小部件的更多细节，请查看 通过部件库（Widget Factory）扩展小部件（Widget）。

清理

在某些情况下，允许用户应用插件，然后再取消应用。您可以通过 `_destroy()` 方法做到这一点。在 `_destroy()` 方法内，您应该撤销在初始化和后期使用期间插件所做的一切动作。 `_destroy()` 是通过 `.destroy()` 方法被调用的， `.destroy()` 方法是在插件实例绑定的元素从 DOM 上移除时被自动调用的，所以这可被用于垃圾回收。基本的 `.destroy()` 方法也处理一些常用的清理操作，比如从小部件的 DOM 元素上移除实例引用，从元素上解除绑定小部件命名空间中的所有事件，解除绑定所有使用 `_bind()` 添加的事件。

```
$.widget( "custom.progressbar", {
    options: {
        value: 0
    },
    _create: function() {
        this.options.value = this._constrain(this.options.value);
        this.element.addClass( "progressbar" );
        this.refresh();
    },
    _setOption: function( key, value ) {
        if ( key === "value" ) {
            value = this._constrain( value );
        }
        this._super( key, value );
    },
    _setOptions: function( options ) {
        this._super( options );
        this.refresh();
    },
    refresh: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
        if ( this.options.value == 100 ) {
            this._trigger( "complete", null, { value: 100 } );
        }
    },
    _constrain: function( value ) {
        if ( value > 100 ) {
            value = 100;
        }
        if ( value < 0 ) {
            value = 0;
        }
        return value;
    },
    _destroy: function() {
        this.element
            .removeClass( "progressbar" )
            .text( "" );
    }
});
```

关闭注释

部件库（Widget Factory）只是创建有状态插件的一种方式。这里还有一些其他不同的模型可以使用，且每个都有各自的优势和劣势。部件库（Widget Factory）解决了很多常见的问题，且大大提高了效率，同时也大大提高了代码的重用性，使它适合于 jQuery UI 及其他有状态的插件。

请注意，在本章节中我们使用了 `custom` 命名空间。 `ui` 命名空间被官方的 jQuery UI 插件保留。当创建您自己的插件时，您应该创建自己的命名空间。这样才能更清楚插件来自哪里，属于哪个范围。

jQuery UI 实例

jQuery UI 实例

jQuery UI 提供了一组用户界面交互、特效、小部件、实用工具及主题。通过实例演示、查看源代码、创建主题、阅读API 文档，开始学习使用 jQuery UI。

交互（Interactions）

交互（Interactions）向任何元素添加基本的基于鼠标的行为。您可以只通过短短的几行代码即可创建排序列表、缩放元素、拖放行为等等。交互（Interactions）也作用于更复杂的小部件和应用程序。

- 拖动（Draggable）
- 放置（Droppable）
- 缩放（Resizable）
- 选择（Selectable）
- 排序（Sortable）

小部件（Widgets）

小部件（Widgets）有功能齐全的 UI 控件，使桌面应用程序也具备 Web 应用程序一样丰富的功能。所有的小部件（Widgets）提供了一个核心，带有定制行为的大量扩展以及完整的主题支持。

- 折叠面板（Accordion）
- 自动完成（Autocomplete）
- 按钮（Button）
- 日期选择器（Datepicker）
- 对话框（Dialog）
- 菜单（Menu）
- 进度条（Progressbar）
- 滑块（Slider）
- 旋转器（Spinner）
- 标签页（Tabs）
- 工具提示框（Tooltip）

特效（Effects）

特效（Effects）支持颜色动画和 Class 转换，同时也提供了一些额外的 Easings。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

- 特效（Effect）
- 可见性（Visibility）

- 显示（Show）
    - 隐藏（Hide）
    - 切换（Toggle）
- Class 动画（Class Animation）
    - 添加 Class（Add Class）
    - 移除 Class（Remove Class）
    - 切换 Class（Toggle Class）
    - 转换 Class（Switch Class）
- 颜色动画（Color Animation）

实用工具（Utilities）

jQuery UI 使用实体工具（Utilities）来创建交互（interactions）和小部件（widgets）。

- 定位（Position）
- 部件库（Widget Factory）

jQuery UI 实例 - 拖动（Draggable）

允许使用鼠标移动元素。

如需了解更多有关 draggable 交互的细节，请查看 API 文档 可拖拽小部件（Draggable Widget）。

默认功能

在任意的 DOM 元素上启用 draggable 功能。通过鼠标点击并在视区中拖动来移动 draggable 对象。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 150px; height: 150px; padding: 0.5em; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>请拖动我！</p>
</div>

</body>
</html>
```

自动滚动

当 draggable 移动到视区外时自动滚动文档。设置 `scroll` 选项为 true 来启用自动滚动，当滚动触发时进行微调，滚动速度是通过 `scrollSensitivity` 和 `scrollSpeed` 选项设置的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 自动滚动</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2, #draggable3 { width: 100px; height: 100p
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ scroll: true });
    $( "#draggable2" ).draggable({ scroll: true, scrollSensitivity
    $( "#draggable3" ).draggable({ scroll: true, scrollSpeed: 100 }
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>Scroll 设置为 true，默认设置</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>scrollSensitivity 设置为 100</p>
</div>

<div id="draggable3" class="ui-widget-content">
  <p>scrollSpeed 设置为 100</p>
</div>

<div style="height: 5000px; width: 1px;"></div>

</body>
</html>
```

约束运动

通过定义 draggable 区域的边界来约束每个 draggable 的运动。设置 `axis` 选项来限制 draggable 的路径为 x 轴或者 y 轴，或者使用 `containment` 选项来指定一个父级的 DOM 元素或者一个 jQuery 选择器，比如 'document.'。

```
<!doctype html>
<html lang="en">
<head>
```

```
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 约束运动</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .draggable { width: 90px; height: 90px; padding: 0.5em; float: le
  #draggable, #draggable2 { margin-bottom:20px; }
  #draggable { cursor: n-resize; }
  #draggable2 { cursor: e-resize; }
  #containment-wrapper { width: 95%; height:150px; border:2px solid
  h3 { clear: left; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ axis: "y" });
    $( "#draggable2" ).draggable({ axis: "x" });

    $( "#draggable3" ).draggable({ containment: "#containment-wrapp
    $( "#draggable5" ).draggable({ containment: "parent" });
  });
  </script>
</head>
<body>

<h3>沿着轴约束运动：</h3>

<div id="draggable" class="draggable ui-widget-content">
  <p>只能垂直拖拽</p>
</div>

<div id="draggable2" class="draggable ui-widget-content">
  <p>只能水平拖拽</p>
</div>

<h3>或者在另一个 DOM 元素中约束运动：</h3>
<div id="containment-wrapper">
  <div id="draggable3" class="draggable ui-widget-content">
    <p>我被约束在盒子里</p>
  </div>

  <div class="draggable ui-widget-content">
    <p id="draggable5" class="ui-widget-header">我被约束在父元素内</p>
  </div>
</div>

</body>
</html>
```

光标样式

当拖拽对象时定位光标。默认情况下，光标是出现在被拖拽对象的中间。使用 `cursorAt` 选项来指定相对于 draggable 的另一个位置（指定一个相对于 top、right、bottom、left 的像素值）。通过提供一个带有有效的 CSS 光标值的 `cursor` 选项，来自定义光标的外观。有效的 CSS 光标值包括：default、move、pointer、crosshair，等等。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 光标样式</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2, #draggable3 { width: 100px; height: 100p
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ cursor: "move", cursorAt: { top:
    $( "#draggable2" ).draggable({ cursor: "crosshair", cursorAt: {
    $( "#draggable3" ).draggable({ cursorAt: { bottom: 0 } });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>我总是在中间（相对于鼠标）</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>我的光标是在 left -5 和 top -5</p>
</div>

<div id="draggable3" class="ui-widget-content">
  <p>我的光标位置只控制了 'bottom' 值</p>
</div>

</body>
</html>
```

延迟开始

通过 `delay` 选项设置延迟开始拖拽的毫秒数。通过 `distance` 选项设置光标被按下且拖拽指定像素后才允许拖拽。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 延迟开始</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 120px; height: 120px; padding: 0
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ distance: 20 });
    $( "#draggable2" ).draggable({ delay: 1000 });
    $( ".ui-draggable" ).disableSelection();
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>只有把我拖拽了 20 像素后，拖拽才开始</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>不管 distance 是多少，您都必须拖拽并等待 1000ms 后拖拽才开始</p>
</div>

</body>
</html>
```

事件

draggable 上的 `start` 、 `drag` 和 `stop` 事件。拖拽开始时触发 `start` 事件，拖拽期间触发 `drag` 事件，拖拽停止时触发 `stop` 事件。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 事件</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 16em; padding: 0 1em; }
  #draggable ul li { margin: 1em 0; padding: 0.5em 0; } * html #dra
  #draggable ul li span.ui-icon { float: left; }
  #draggable ul li span.count { font-weight: bold; }
  </style>
  <script>
  $(function() {
    var $start_counter = $( "#event-start" ),
      $drag_counter = $( "#event-drag" ),
      $stop_counter = $( "#event-stop" ),
      counts = [ 0, 0, 0 ];

    $( "#draggable" ).draggable({
      start: function() {
        counts[ 0 ]++;
        updateCounterStatus( $start_counter, counts[ 0 ] );
      },
      drag: function() {
        counts[ 1 ]++;
        updateCounterStatus( $drag_counter, counts[ 1 ] );
      },
      stop: function() {
        counts[ 2 ]++;
        updateCounterStatus( $stop_counter, counts[ 2 ] );
      }
    });

    function updateCounterStatus( $event_counter, new_count ) {
      // 首先更新视觉状态...
      if ( !$event_counter.hasClass( "ui-state-hover" ) ) {
        $event_counter.addClass( "ui-state-hover" )
          .siblings().removeClass( "ui-state-hover" );
      }
      // ...然后更新数字
      $( "span.count", $event_counter ).text( new_count );
    }
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget ui-widget-content">

  <p>请拖拽我，触发一连串的事件。</p>

  <ul class="ui-helper-reset">
    <li id="event-start" class="ui-state-default ui-corner-all"><sp
    <li id="event-drag" class="ui-state-default ui-corner-all"><spa
    <li id="event-stop" class="ui-state-default ui-corner-all"><spa
  </ul>
```

```
</div>

</body>
</html>
```

Handles

只有当光标在 draggable 上指定部分时才允许拖拽。使用 `handle` 选项来指定用于拖拽对象的元素（或元素组）的 jQuery 选择器。

Or prevent dragging when the cursor is over a specific element (or group of elements) within the draggable. Use the cancel option to specify a jQuery selector over which to "cancel" draggable functionality.

或者当光标在 draggable 内指定元素（或元素组）上时不允许拖拽。使用 `handle` 选项来指定取消拖拽功能的 jQuery 选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - Handles</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 100px; height: 100px; padding: 0
  #draggable p { cursor: move; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ handle: "p" });
    $( "#draggable2" ).draggable({ cancel: "p.ui-widget-header" });
    $( "div, p" ).disableSelection();
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p class="ui-widget-header">您只可以在这个范围内拖拽我</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>您可以把我向四周拖拽&hellip;</p>
  <p class="ui-widget-header">&hellip;但是您不可以在这个范围内拖拽我</p>
</div>

</body>
</html>
```

还原位置

当带有布尔值 `revert` 选项的 draggable 停止拖拽时，返回 draggable（或它的助手）到原始位置。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 还原位置</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 100px; height: 100px; padding: 0
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ revert: true });
    $( "#draggable2" ).draggable({ revert: true, helper: "clone" })
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>还原</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>还原助手</p>
</div>

</body>
</html>
```

对齐到元素或网格

对齐 draggable 到 DOM 元素的内部或外部边界。使用 `snap` 、 `snapMode` （inner, outer, both）和 `snapTolerance` （当调用对齐时，draggable 与元素之间的距离，以像素为单位）选项。

或者对齐 draggable 到网格。通过 `grid` 选项设置网格单元的尺寸（以像素为单位的高度或宽度）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 对齐到元素或网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .draggable { width: 90px; height: 80px; padding: 5px; float: left
  .ui-widget-header p, .ui-widget-content p { margin: 0; }
  #snaptarget { height: 140px; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ snap: true });
    $( "#draggable2" ).draggable({ snap: ".ui-widget-header" });
    $( "#draggable3" ).draggable({ snap: ".ui-widget-header", snapM
    $( "#draggable4" ).draggable({ grid: [ 20, 20 ] });
    $( "#draggable5" ).draggable({ grid: [ 80, 80 ] });
  });
  </script>
</head>
<body>

<div id="snaptarget" class="ui-widget-header">
  <p>我是对齐目标</p>
</div>

<br style="clear:both">

<div id="draggable" class="draggable ui-widget-content">
  <p>默认（snap: true），对齐到所有其他的 draggable 元素</p>
</div>

<div id="draggable2" class="draggable ui-widget-content">
  <p>我只对齐到大盒子</p>
</div>

<div id="draggable3" class="draggable ui-widget-content">
  <p>我只对齐到大盒子的外边缘</p>
</div>

<div id="draggable4" class="draggable ui-widget-content">
  <p>我对齐到一个 20 x 20 网格</p>
</div>

<div id="draggable5" class="draggable ui-widget-content">
  <p>我对齐到一个 80 x 80 网格</p>
</div>

</body>
</html>
```

视觉反馈

给用户提供反馈，就像以助手方式拖拽对象一样。 `helper` 选项接受值 'original'（用光标移动 draggable 对象），'clone'（用光标移动 draggable 的副本），或者一个返回 DOM 元素的函数（该元素在拖拽期间显示在光标附近）。通过 `opacity` 选项控制助手的透明度。

为了区别哪一个 draggable 正在被拖拽，让在运动中的 draggable 保持最前。如果正在拖拽，使用 `zIndex` 选项来设置助手的高度 z-index，或者使用 `stack` 选项来确保当停止拖拽时，最后一个被拖拽的项目总是出现在同组其他项目的上面。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） - 视觉反馈</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2, #draggable3, #set div { width: 90px; hei
  #draggable, #draggable2, #draggable3 { margin-bottom:20px; }
  #set { clear:both; float:left; width: 368px; height: 120px; }
  p { clear:both; margin:0; padding:1em 0; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ helper: "original" });
    $( "#draggable2" ).draggable({ opacity: 0.7, helper: "clone" })
    $( "#draggable3" ).draggable({
      cursor: "move",
      cursorAt: { top: -12, left: -20 },
      helper: function( event ) {
        return $( "<div class='ui-widget-header'>I'm a custom helpe
      }
    });
    $( "#set div" ).draggable({ stack: "#set div" });
  });
  </script>
</head>
<body>

<h3 class="docs">助手：</h3>

<div id="draggable" class="ui-widget-content">
  <p>原始的</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>半透明的克隆</p>
</div>

<div id="draggable3" class="ui-widget-content">
```

```
  <p>自定义助手（与 cursorAt 结合）</p>
</div>

<h3 class="docs">堆叠：</h3>
<div id="set">
  <div class="ui-widget-content">
    <p>我们是 draggables..</p>
  </div>

  <div class="ui-widget-content">
    <p>..它的 z-index 是自动控制的..</p>
  </div>

  <div class="ui-widget-content">
    <p>..带有 stack 选项。</p>
  </div>
</div>

</body>
</html>
```

jQuery UI Draggable + Sortable

Draggable 与 Sortable 的无缝交互。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 拖动（Draggable） + 排序（Sortable）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  ul { list-style-type: none; margin: 0; padding: 0; margin-bottom
  li { margin: 5px; padding: 5px; width: 150px; }
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable({
      revert: true
    });
    $( "#draggable" ).draggable({
      connectToSortable: "#sortable",
      helper: "clone",
      revert: "invalid"
    });
    $( "ul, li" ).disableSelection();
  });
  </script>
</head>
<body>

<ul>
  <li id="draggable" class="ui-state-highlight">请拖拽我</li>
</ul>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
</ul>

</body>
</html>
```

jQuery UI 实例 - 放置（Droppable）

为可拖拽小部件创建目标。

如需了解更多有关 droppable 交互的细节，请查看 API 文档 可放置小部件（Droppable Widget）。

默认功能

在任意的 DOM 元素上启用 droppable 功能，并为可拖拽小部件创建目标。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 100px; height: 100px; padding: 0.5em; float:
  #droppable { width: 150px; height: 150px; padding: 0.5em; float:
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();
    $( "#droppable" ).droppable({
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>请把我拖拽到目标处！</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>请放置在这里！</p>
</div>

</body>
</html>
```

接受

使用 `accept` 选项指定目标 droppable 接受哪一个元素（或元素组）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 接受</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #droppable { width: 150px; height: 150px; padding: 0.5em; float:
  #draggable, #draggable-nonvalid { width: 100px; height: 100px; pa
  </style>
  <script>
  $(function() {
    $( "#draggable, #draggable-nonvalid" ).draggable();
    $( "#droppable" ).droppable({
      accept: "#draggable",
      activeClass: "ui-state-hover",
      hoverClass: "ui-state-active",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable-nonvalid" class="ui-widget-content">
  <p>我是不能被放置的 draggable</p>
</div>

<div id="draggable" class="ui-widget-content">
  <p>请拖拽我到目标</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>accept: '#draggable'</p>
</div>

</body>
</html>
```

防止传播

当使用嵌套的 droppable 时 — 例如，您可以有一个树形的可编辑的目录结构，带有文件夹和文档节点 — `greedy` 选项设置为 true 来防止当 draggable 被放置在子节点（droppable）上时的事件传播。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 防止传播</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable { width: 100px; height: 40px; padding: 0.5em; float: l
  #droppable, #droppable2 { width: 230px; height: 120px; padding: 0
  #droppable-inner, #droppable2-inner { width: 170px; height: 60px;
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();

    $( "#droppable, #droppable-inner" ).droppable({
      activeClass: "ui-state-hover",
      hoverClass: "ui-state-active",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "> p" )
            .html( "Dropped!" );
        return false;
      }
    });

    $( "#droppable2, #droppable2-inner" ).droppable({
      greedy: true,
      activeClass: "ui-state-hover",
      hoverClass: "ui-state-active",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "> p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>请拖拽我到目标</p>
```

```
</div>

<div id="droppable" class="ui-widget-header">
  <p>外部 droppable</p>
  <div id="droppable-inner" class="ui-widget-header">
    <p>内部 droppable（不带有 greedy）</p>
  </div>
</div>

<div id="droppable2" class="ui-widget-header">
  <p>外部 droppable</p>
  <div id="droppable2-inner" class="ui-widget-header">
    <p>内部 droppable（带有 greedy）</p>
  </div>
</div>

</body>
</html>
```

还原 draggable 的位置

当带有布尔值 `revert` 选项的 draggable 停止拖拽时，返回 draggable（或它的助手）到原始位置。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 还原 draggable 的位置</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 100px; height: 100px; padding: 0
  #droppable { width: 150px; height: 150px; padding: 0.5em; float:
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable({ revert: "valid" });
    $( "#draggable2" ).draggable({ revert: "invalid" });

    $( "#droppable" ).droppable({
      activeClass: "ui-state-default",
      hoverClass: "ui-state-hover",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "已放置！" );
      }
    });
  });
  </script>
</head>
<body>

<div id="draggable" class="ui-widget-content">
  <p>当被放置在目标上时还原</p>
</div>

<div id="draggable2" class="ui-widget-content">
  <p>当未被放置在目标上时还原</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>请放置在这里</p>
</div>

</body>
</html>
```

购物车演示

演示如何使用折叠面板来展示产品的目录结构，使用拖拽和放置来添加产品到购物车，购物车中的产品是可排序的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 购物车演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  h1 { padding: .2em; margin: 0; }
  #products { float:left; width: 500px; margin-right: 2em; }
  #cart { width: 200px; float: left; margin-top: 1em; }
  /* 定义列表样式，以便最大化 droppable */
  #cart ol { margin: 0; padding: 1em 0 1em 3em; }
  </style>
  <script>
  $(function() {
    $( "#catalog" ).accordion();
    $( "#catalog li" ).draggable({
      appendTo: "body",
      helper: "clone"
    });
    $( "#cart ol" ).droppable({
      activeClass: "ui-state-default",
      hoverClass: "ui-state-hover",
      accept: ":not(.ui-sortable-helper)",
      drop: function( event, ui ) {
        $( this ).find( ".placeholder" ).remove();
        $( "<li></li>" ).text( ui.draggable.text() ).appendTo( this
      }
    }).sortable({
      items: "li:not(.placeholder)",
      sort: function() {
        // 获取由 droppable 与 sortable 交互而加入的条目
        // 使用 connectWithSortable 可以解决这个问题，但不允许您自定义 ac
        $( this ).removeClass( "ui-state-default" );
      }
    });
  });
  </script>
</head>
<body>

<div id="products">
  <h1 class="ui-widget-header">产品</h1>
  <div id="catalog">
    <h2><a href="#">T-Shirts</a></h2>
    <div>
```

```
      <ul>
        <li>Lolcat Shirt</li>
        <li>Cheezeburger Shirt</li>
        <li>Buckit Shirt</li>
      </ul>
    </div>
    <h2><a href="#">Bags</a></h2>
    <div>
      <ul>
        <li>Zebra Striped</li>
        <li>Black Leather</li>
        <li>Alligator Leather</li>
      </ul>
    </div>
    <h2><a href="#">Gadgets</a></h2>
    <div>
      <ul>
        <li>iPhone</li>
        <li>iPod</li>
        <li>iPad</li>
      </ul>
    </div>
  </div>
</div>

<div id="cart">
  <h1 class="ui-widget-header">购物车</h1>
  <div class="ui-widget-content">
    <ol>
      <li class="placeholder">添加产品到这里</li>
    </ol>
  </div>
</div>

</body>
</html>
```

简单的照片管理器

您可以通过拖拽照片到回收站或者点击回收站图标来删除照片。

您可以通过拖拽照片到相册或者点击回收利用图标来还原照片。

您可以通过点击缩放图标来查看大图。jQuery UI 对话框（dialog）部件用于模态窗口。

```
<!doctype html>
<html lang="en">
<head>
```

```
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 简单的照片管理器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #gallery { float: left; width: 65%; min-height: 12em; }
  .gallery.custom-state-active { background: #eee; }
  .gallery li { float: left; width: 96px; padding: 0.4em; margin: 0
  .gallery li h5 { margin: 0 0 0.4em; cursor: move; }
  .gallery li a { float: right; }
  .gallery li a.ui-icon-zoomin { float: left; }
  .gallery li img { width: 100%; cursor: move; }

  #trash { float: right; width: 32%; min-height: 18em; padding: 1%;
  #trash h4 { line-height: 16px; margin: 0 0 0.4em; }
  #trash h4 .ui-icon { float: left; }
  #trash .gallery h5 { display: none; }
  </style>
  <script>
  $(function() {
    // 这是相册和回收站
    var $gallery = $( "#gallery" ),
      $trash = $( "#trash" );

    // 让相册的条目可拖拽
    $( "li", $gallery ).draggable({
      cancel: "a.ui-icon", // 点击一个图标不会启动拖拽
      revert: "invalid", // 当未被放置时，条目会还原回它的初始位置
      containment: "document",
      helper: "clone",
      cursor: "move"
    });

    // 让回收站可放置，接受相册的条目
    $trash.droppable({
      accept: "#gallery > li",
      activeClass: "ui-state-highlight",
      drop: function( event, ui ) {
        deleteImage( ui.draggable );
      }
    });

    // 让相册可放置，接受回收站的条目
    $gallery.droppable({
      accept: "#trash li",
      activeClass: "custom-state-active",
      drop: function( event, ui ) {
        recycleImage( ui.draggable );
      }
    });
```

```
    // 图像删除功能
    var recycle_icon = "<a href='link/to/recycle/script/when/we/hav
    function deleteImage( $item ) {
      $item.fadeOut(function() {
        var $list = $( "ul", $trash ).length ?
          $( "ul", $trash ) :
          $( "<ul class='gallery ui-helper-reset'/>" ).appendTo( $t

        $item.find( "a.ui-icon-trash" ).remove();
        $item.append( recycle_icon ).appendTo( $list ).fadeIn(funct
          $item
            .animate({ width: "48px" })
            .find( "img" )
              .animate({ height: "36px" });
        });
      });
    }

    // 图像还原功能
    var trash_icon = "<a href='link/to/trash/script/when/we/have/js
    function recycleImage( $item ) {
      $item.fadeOut(function() {
        $item
          .find( "a.ui-icon-refresh" )
            .remove()
          .end()
          .css( "width", "96px")
          .append( trash_icon )
          .find( "img" )
            .css( "height", "72px" )
          .end()
          .appendTo( $gallery )
          .fadeIn();
      });
    }

    // 图像预览功能，演示 ui.dialog 作为模态窗口使用
    function viewLargerImage( $link ) {
      var src = $link.attr( "href" ),
        title = $link.siblings( "img" ).attr( "alt" ),
        $modal = $( "img[src$='" + src + "']" );

      if ( $modal.length ) {
        $modal.dialog( "open" );
      } else {
        var img = $( "<img alt='" + title + "' width='384' height=
          .attr( "src", src ).appendTo( "body" );
        setTimeout(function() {
          img.dialog({
            title: title,
            width: 400,
            modal: true
          });
```

```
        }, 1 );
      }
    }

    // 通过事件代理解决图标行为
    $( "ul.gallery > li" ).click(function( event ) {
      var $item = $( this ),
        $target = $( event.target );

      if ( $target.is( "a.ui-icon-trash" ) ) {
        deleteImage( $item );
      } else if ( $target.is( "a.ui-icon-zoomin" ) ) {
        viewLargerImage( $target );
      } else if ( $target.is( "a.ui-icon-refresh" ) ) {
        recycleImage( $item );
      }

      return false;
    });
  });
  </script>
</head>
<body>

<div class="ui-widget ui-helper-clearfix">

<ul id="gallery" class="gallery ui-helper-reset ui-helper-clearfix'
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras</h5>
    <img src="img/high_tatras_min.jpg" alt="High Tatras 的最高峰" wi
    <a href="img/high_tatras.jpg" title="查看大图" class="ui-icon ui
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras 2</h5>
    <img src="img/high_tatras2_min.jpg" alt="绿山湖畔的小屋" width="9
    <a href="img/high_tatras2.jpg" title="查看大图" class="ui-icon u
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras 3</h5>
    <img src="img/high_tatras3_min.jpg" alt="计划登高" width="96" he
    <a href="img/high_tatras3.jpg" title="查看大图" class="ui-icon u
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
  <li class="ui-widget-content ui-corner-tr">
    <h5 class="ui-widget-header">High Tatras 4</h5>
    <img src="img/high_tatras4_min.jpg" alt="在 Kozi kopka 的顶部" w
    <a href="img/high_tatras4.jpg" title="查看大图" class="ui-icon u
    <a href="link/to/trash/script/when/we/have/js/off" title="删除图
  </li>
</ul>
```

```
<div id="trash" class="ui-widget-content ui-state-default">
  <h4 class="ui-widget-header"><span class="ui-icon ui-icon-trash">
</div>

</div>

</body>
</html>
```

视觉反馈

当悬停在 droppable 上时，或者当 droppable 被激活（即一个可接受的 draggable 被放置在 droppable 上）时，改变 droppable 的外观。使用 `hoverClass` 或 `activeClass` 选项来分别指定 class。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 放置（Droppable） - 视觉反馈</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #draggable, #draggable2 { width: 90px; height: 90px; padding: 0.5
  #droppable, #droppable2 { width: 120px; height: 120px; padding: 0
  h3 { clear: left; }
  </style>
  <script>
  $(function() {
    $( "#draggable" ).draggable();
    $( "#droppable" ).droppable({
      hoverClass: "ui-state-hover",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
          .find( "p" )
            .html( "Dropped!" );
      }
    });

    $( "#draggable2" ).draggable();
    $( "#droppable2" ).droppable({
      accept: "#draggable2",
      activeClass: "ui-state-default",
      drop: function( event, ui ) {
        $( this )
          .addClass( "ui-state-highlight" )
```

```
          .find( "p" )
            .html( "Dropped!" );
      }
    });
  });
  </script>
</head>
<body>

<h3>当悬停在 droppable 上时的反馈：</h3>

<div id="draggable" class="ui-widget-content">
  <p>请拖拽我到目标</p>
</div>

<div id="droppable" class="ui-widget-header">
  <p>请放置在这里</p>
</div>

<h3>当激活 draggable 时的反馈：</h3>

<div id="draggable2" class="ui-widget-content">
  <p>请拖拽我到目标</p>
</div>

<div id="droppable2" class="ui-widget-header">
  <p>请放置在这里</p>
</div>

</body>
</html>
```

jQuery UI 实例 - 缩放（Resizable）

使用鼠标改变元素的尺寸。

如需了解更多有关 resizable 交互的细节，请查看 API 文档 可调整尺寸小部件（Resizable Widget）。

默认功能

在任意的 DOM 元素上启用 resizable 功能。通过鼠标拖拽右边或底边的边框到所需的宽度或高度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable();
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">缩放（Resizable）</h3>
</div>

</body>
</html>
```

动画

使用 `animate` 选项（布尔值）使缩放行为动画化。当该选项设置为 true 时，拖拽轮廓到所需的位置，元素会在拖拽停止时以动画形式调整到该尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 动画</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-helper { border: 1px dotted gray; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      animate: true
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">动画</h3>
</div>

</body>
</html>
```

限制缩放区域

定义缩放区域的边界。使用 `containment` 选项来指定一个父级的 DOM 元素或一个 jQuery 选择器，比如 'document.'。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 限制缩放区域</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #container { width: 300px; height: 300px; }
  #container h3 { text-align: center; margin: 0; margin-bottom: 10p
  #resizable { background-position: top left; width: 150px; height
  #resizable, #container { padding: 0.5em; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      containment: "#container"
    });
  });
  </script>
</head>
<body>

<div id="container" class="ui-widget-content">
  <h3 class="ui-widget-header">限制</h3>
  <div id="resizable" class="ui-state-active">
    <h3 class="ui-widget-header">缩放（Resizable）</h3>
  </div>
</div>

</body>
</html>
```

延迟开始

通过 `delay` 选项设置延迟开始缩放的毫秒数。通过 `distance` 选项设置光标被按下且拖拽指定像素后才允许缩放。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 延迟开始</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable, #resizable2 { width: 150px; height: 150px; padding: (
  #resizable h3, #resizable2 h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      delay: 1000
    });

    $( "#resizable2" ).resizable({
      distance: 40
    });
  });
  </script>
</head>
<body>

<h3 class="docs">时间延迟 (ms):</h3>
<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">时间</h3>
</div>

<h3 class="docs">距离延迟 (px):</h3>
<div id="resizable2" class="ui-widget-content">
  <h3 class="ui-widget-header">距离</h3>
</div>

</body>
</html>
```

助手

通过设置 `helper` 选项为一个 CSS class，当缩放时只显示元素的轮廓。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 助手</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-helper { border: 2px dotted #00F; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      helper: "ui-resizable-helper"
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">助手</h3>
</div>

</body>
</html>
```

最大/最小尺寸

使用 `maxHeight`、`maxWidth`、`minHeight` 和 `minWidth` 选项限制 resizable 元素的最大或最小高度或宽度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 最大/最小尺寸</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 200px; height: 150px; padding: 5px; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      maxHeight: 250,
      maxWidth: 350,
      minHeight: 150,
      minWidth: 200
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">放大/缩小</h3>
</div>

</body>
</html>
```

保持纵横比

保持现有的纵横比或设置一个新的纵横比来限制缩放比例。设置 `aspectRatio` 选项为 true，且可选地传递一个新的比率（比如，4/3）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 保持纵横比</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 160px; height: 90px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      aspectRatio: 16 / 9
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">保持纵横比</h3>
</div>

</body>
</html>
```

对齐到网格

对齐 resizable 元素到网格。通过 `grid` 选项设置网格单元的尺寸（以像素为单位的高度和宽度）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 对齐到网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      grid: 50
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">网格</h3>
</div>

</body>
</html>
```

同步缩放

通过点击并拖拽一个元素的边来同时调整多个元素的尺寸。给 `alsoResize` 选项传递一个共享的选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 同步缩放</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { background-position: top left; }
  #resizable, #also { width: 150px; height: 120px; padding: 0.5em;
  #resizable h3, #also h3 { text-align: center; margin: 0; }
  #also { margin-top: 1em; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      alsoResize: "#also"
    });
    $( "#also" ).resizable();
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-header">
  <h3 class="ui-state-active">缩放</h3>
</div>

<div id="also" class="ui-widget-content">
  <h3 class="ui-widget-header">同步缩放</h3>
</div>

</body>
</html>
```

文本框

可缩放的文本框。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 文本框</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-resizable-se {
    bottom: 17px;
  }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      handles: "se"
    });
  });
  </script>
</head>
<body>

<textarea id="resizable" rows="5" cols="20"></textarea>

</body>
</html>
```

视觉反馈

通过设置 `ghost` 选项为 true，可在缩放期间显示一个半透明的元素，而不是显示一个实际的元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 缩放（Resizable） - 视觉反馈</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-ghost { border: 1px dotted gray; }
  </style>
  <script>
  $(function() {
    $( "#resizable" ).resizable({
      ghost: true
    });
  });
  </script>
</head>
<body>

<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">Ghost</h3>
</div>

</body>
</html>
```

jQuery UI 实例 - 选择（Selectable）

使用鼠标选择单个元素或一组元素。

如需了解更多有关 selectable 交互的细节，请查看 API 文档 可选择小部件（Selectable Widget）。

默认功能

在某个 DOM 元素上或者一组元素上启用 selectable 功能。通过鼠标拖拽选择条目。按住 Ctrl 键，选择多个不相邻的条目。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 选择（Selectable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/

  <style>
  #feedback { font-size: 1.4em; }
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; widtl
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; l
  </style>
  <script>
  $(function() {
    $( "#selectable" ).selectable();
  });
  </script>
</head>
<body>

<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
  <li class="ui-widget-content">Item 6</li>
  <li class="ui-widget-content">Item 7</li>
</ol>

</body>
</html>
```

显示为网格

让 selectable 条目显示为网格，使用 CSS 使得它们带有相同的尺寸且浮动显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 选择（Selectable） - 显示为网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/

  <style>
  #feedback { font-size: 1.4em; }
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; widt
  #selectable li { margin: 3px; padding: 1px; float: left; width: 1
  </style>
  <script>
  $(function() {
    $( "#selectable" ).selectable();
  });
  </script>
</head>
<body>

<ol id="selectable">
  <li class="ui-state-default">1</li>
  <li class="ui-state-default">2</li>
  <li class="ui-state-default">3</li>
  <li class="ui-state-default">4</li>
  <li class="ui-state-default">5</li>
  <li class="ui-state-default">6</li>
  <li class="ui-state-default">7</li>
  <li class="ui-state-default">8</li>
  <li class="ui-state-default">9</li>
  <li class="ui-state-default">10</li>
  <li class="ui-state-default">11</li>
  <li class="ui-state-default">12</li>
</ol>

</body>
</html>
```

序列化

写一个函数，在 `stop` 事件发生时触发，来收集被选中条目的索引值。呈现这些值作为反馈，或者以数据字符串形式进行传递。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 选择（Selectable） - 序列化</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/

  <style>
  #feedback { font-size: 1.4em; }
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; widtl
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; l
  </style>
  <script>
  $(function() {
    $( "#selectable" ).selectable({
      stop: function() {
        var result = $( "#select-result" ).empty();
        $( ".ui-selected", this ).each(function() {
          var index = $( "#selectable li" ).index( this );
          result.append( " #" + ( index + 1 ) );
        });
      }
    });
  });
  </script>
</head>
<body>

<p id="feedback">
<span>您已经选择了：</span> <span id="select-result">无</span>。
</p>

<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
  <li class="ui-widget-content">Item 6</li>
</ol>

</body>
</html>
```

jQuery UI 实例 - 排序（Sortable）

使用鼠标调整列表中或者网格中元素的排序。

如需了解更多有关 sortable 交互的细节，请查看 API 文档 可排序小部件（Sortable Widget）。

默认功能

在任意的 DOM 元素上启用 sortable 功能。通过鼠标点击并拖拽元素到列表中的一个新的位置，其它条目会自动调整。默认情况下，sortable 各个条目共享 `draggable` 属性。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable { list-style-type: none; margin: 0; padding: 0; width: 
  #sortable li { margin: 0 3px 3px 3px; padding: 0.4em; padding-lef
  #sortable li span { position: absolute; margin-left: -1.3em; }
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable();
    $( "#sortable" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
  <li class="ui-state-default"><span class="ui-icon ui-icon-arrowth
</ul>

</body>
</html>
```

连接列表

通过向 `connectWith` 选项传递一个选择器，来把一个列表中的元素排序到另一个列表中，反之亦然。最简单的办法是将带有某个 CSS class 的所有相关的列表分组，然后传递该 class 到 sortable 函数（比如， `connectWith: '.myclass'` ）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 连接列表</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2 { list-style-type: none; margin: 0; paddin
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5p
  </style>
  <script>
  $(function() {
    $( "#sortable1, #sortable2" ).sortable({
      connectWith: ".connectedSortable"
    }).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable1" class="connectedSortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
</ul>

<ul id="sortable2" class="connectedSortable">
  <li class="ui-state-highlight">Item 1</li>
  <li class="ui-state-highlight">Item 2</li>
  <li class="ui-state-highlight">Item 3</li>
  <li class="ui-state-highlight">Item 4</li>
  <li class="ui-state-highlight">Item 5</li>
</ul>

</body>
</html>
```

标签页连接列表

通过放置列表项到顶部适当的标签页中，来把一个列表中的元素排序到另一个列表中，反之亦然。

```
<!doctype html>
```

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 标签页连接列表</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5p
  </style>
  <script>
  $(function() {
    $( "#sortable1, #sortable2" ).sortable().disableSelection();

    var $tabs = $( "#tabs" ).tabs();

    var $tab_items = $( "ul:first li", $tabs ).droppable({
      accept: ".connectedSortable li",
      hoverClass: "ui-state-hover",
      drop: function( event, ui ) {
        var $item = $( this );
        var $list = $( $item.find( "a" ).attr( "href" ) )
          .find( ".connectedSortable" );

        ui.draggable.hide( "slow", function() {
          $tabs.tabs( "option", "active", $tab_items.index( $item )
          $( this ).appendTo( $list ).show( "slow" );
        });
      }
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
  </ul>
  <div id="tabs-1">
    <ul id="sortable1" class="connectedSortable ui-helper-reset">
      <li class="ui-state-default">Item 1</li>
      <li class="ui-state-default">Item 2</li>
      <li class="ui-state-default">Item 3</li>
      <li class="ui-state-default">Item 4</li>
      <li class="ui-state-default">Item 5</li>
    </ul>
  </div>
  <div id="tabs-2">
    <ul id="sortable2" class="connectedSortable ui-helper-reset">
      <li class="ui-state-highlight">Item 1</li>
```

```
        <li class="ui-state-highlight">Item 2</li>
        <li class="ui-state-highlight">Item 3</li>
        <li class="ui-state-highlight">Item 4</li>
        <li class="ui-state-highlight">Item 5</li>
      </ul>
    </div>
  </div>

</body>
</html>
```

延迟开始

通过时间延迟和距离延迟来防止意外的排序。通过 `delay` 选项设置开始排序之前必须拖拽的毫秒数。通过 `distance` 选项设置开始排序之前必须拖拽的像素数。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 延迟开始</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2 { list-style-type: none; margin: 0; paddin
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5p
  </style>
  <script>
  $(function() {
    $( "#sortable1" ).sortable({
      delay: 300
    });

    $( "#sortable2" ).sortable({
      distance: 15
    });

    $( "li" ).disableSelection();
  });
  </script>
</head>
<body>

<h3 class="docs">时间延迟 300ms：</h3>

<ul id="sortable1">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
</ul>

<h3 class="docs">距离延迟 15px：</h3>

<ul id="sortable2">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
</ul>

</body>
</html>
```

显示为网格

让 sortable 条目显示为网格，使用 CSS 使得它们带有相同的尺寸且浮动显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 显示为网格</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable { list-style-type: none; margin: 0; padding: 0; width:
  #sortable li { margin: 3px 3px 3px 0; padding: 1px; float: left;
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable();
    $( "#sortable" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">1</li>
  <li class="ui-state-default">2</li>
  <li class="ui-state-default">3</li>
  <li class="ui-state-default">4</li>
  <li class="ui-state-default">5</li>
  <li class="ui-state-default">6</li>
  <li class="ui-state-default">7</li>
  <li class="ui-state-default">8</li>
  <li class="ui-state-default">9</li>
  <li class="ui-state-default">10</li>
  <li class="ui-state-default">11</li>
  <li class="ui-state-default">12</li>
</ul>

</body>
</html>
```

放置占位符

当拖拽一个 sortable 条目到一个新的位置时，其他条目将为该条目腾出空间。向 `placeholder` 选项传递一个 class 来定义可视化的空白的样式。使用布尔值的 `forcePlaceholderSize` 选项来设置占位符的尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 放置占位符</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable { list-style-type: none; margin: 0; padding: 0; width: 
  #sortable li { margin: 0 5px 5px 5px; padding: 5px; font-size: 1.
  html>body #sortable li { height: 1.5em; line-height: 1.2em; }
  .ui-state-highlight { height: 1.5em; line-height: 1.2em; }
  </style>
  <script>
  $(function() {
    $( "#sortable" ).sortable({
      placeholder: "ui-state-highlight"
    });
    $( "#sortable" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>

</body>
</html>
```

处理空列表

通过把 选项设置为 `false`，来阻止一个列表中的所有条目被放置到一个单独的空列表中。默认情况下，sortable 条目可被放置到空的列表中。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 处理空列表</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2, #sortable3 { list-style-type: none; margi
  #sortable1 li, #sortable2 li, #sortable3 li { margin: 5px; paddir
  </style>
  <script>
  $(function() {
    $( "ul.droptrue" ).sortable({
      connectWith: "ul"
    });

    $( "ul.dropfalse" ).sortable({
      connectWith: "ul",
      dropOnEmpty: false
    });

    $( "#sortable1, #sortable2, #sortable3" ).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable1" class="droptrue">
  <li class="ui-state-default">可被放置到..</li>
  <li class="ui-state-default">..一个空列表中</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
</ul>

<ul id="sortable2" class="dropfalse">
  <li class="ui-state-highlight">不可被放置到..</li>
  <li class="ui-state-highlight">..一个空列表中</li>
  <li class="ui-state-highlight">Item 3</li>
  <li class="ui-state-highlight">Item 4</li>
  <li class="ui-state-highlight">Item 5</li>
</ul>

<ul id="sortable3" class="droptrue">
</ul>

<br style="clear:both">

</body>
```

```
</html>
```

包含/排除条目

指定通过向 `items` 选项传递一个 jQuery 选择器哪些项目可排序。该选项之外的项目则是不可排序的，同时它们也不是 sortable 条目的有效的目标。

如果只想防止特定的条目排序，则向 `cancel` 选项传递一个 jQuery 选择器。已取消的条目依然是其他条目的有效的排序目标。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 包含/排除条目</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #sortable1, #sortable2 { list-style-type: none; margin: 0; paddin
  #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 3p
  </style>
  <script>
  $(function() {
    $( "#sortable1" ).sortable({
      items: "li:not(.ui-state-disabled)"
    });

    $( "#sortable2" ).sortable({
      cancel: ".ui-state-disabled"
    });

    $( "#sortable1 li, #sortable2 li" ).disableSelection();
  });
  </script>
</head>
<body>

<h3 class="docs">指定哪个条目是 sortable：</h3>

<ul id="sortable1">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default ui-state-disabled">（我不是 sortable 或
  <li class="ui-state-default ui-state-disabled">（我不是 sortable 或
  <li class="ui-state-default">Item 4</li>
</ul>

<h3 class="docs">取消排序（但作为放置目标）：</h3>

<ul id="sortable2">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default ui-state-disabled">（我不是 sortable）
  <li class="ui-state-default ui-state-disabled">（我不是 sortable）
  <li class="ui-state-default">Item 4</li>
</ul>

</body>
</html>
```

门户组件（Portlets）

启用门户组件（样式化的 div）作为 sortable，并使用 `connectWith` 选项来允许在列之间进行排序。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 排序（Sortable） - 门户组件（Portlets）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  body {
    min-width: 520px;
  }
  .column {
    width: 170px;
    float: left;
    padding-bottom: 100px;
  }
  .portlet {
    margin: 0 1em 1em 0;
    padding: 0.3em;
  }
  .portlet-header {
    padding: 0.2em 0.3em;
    margin-bottom: 0.5em;
    position: relative;
  }
  .portlet-toggle {
    position: absolute;
    top: 50%;
    right: 0;
    margin-top: -8px;
  }
  .portlet-content {
    padding: 0.4em;
  }
  .portlet-placeholder {
    border: 1px dotted black;
    margin: 0 1em 1em 0;
    height: 50px;
  }
  </style>
  <script>
  $(function() {
    $( ".column" ).sortable({
      connectWith: ".column",
```

```
      handle: ".portlet-header",
      cancel: ".portlet-toggle",
      placeholder: "portlet-placeholder ui-corner-all"
    });

    $( ".portlet" )
      .addClass( "ui-widget ui-widget-content ui-helper-clearfix ui
      .find( ".portlet-header" )
        .addClass( "ui-widget-header ui-corner-all" )
        .prepend( "<span class='ui-icon ui-icon-minusthick portlet-

    $( ".portlet-toggle" ).click(function() {
      var icon = $( this );
      icon.toggleClass( "ui-icon-minusthick ui-icon-plusthick" );
      icon.closest( ".portlet" ).find( ".portlet-content" ).toggle(
    });
  });
  </script>
</head>
<body>

<div class="column">

  <div class="portlet">
    <div class="portlet-header">订阅</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

  <div class="portlet">
    <div class="portlet-header">新闻</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

</div>

<div class="column">

  <div class="portlet">
    <div class="portlet-header">购物</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

</div>

<div class="column">

  <div class="portlet">
    <div class="portlet-header">链接</div>
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

  <div class="portlet">
    <div class="portlet-header">图像</div>
```

```
    <div class="portlet-content">Lorem ipsum dolor sit amet, consec
  </div>

</div>

</body>
</html>
```

jQuery UI 实例 - 折叠面板（Accordion）

在一个有限的空间内显示用于呈现信息的可折叠的内容面板。

如需了解更多有关 accordion 部件的细节，请查看 API 文档 折叠面板部件（Accordion Widget）。

默认功能

点击头部展开/折叠被分为各个逻辑部分的内容，就像标签页（tabs）一样。您可以选择性地设置当鼠标悬停时是否切换各部分的打开/关闭状态。

基本的 HTML 标记是一系列的标题（H3 标签）和内容 div，因此内容不用通过 JavaScript 即可用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion();
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>
    Mauris mauris ante, blandit et, ultrices a, suscipit eget, quam
    ut neque. Vivamus nisi metus, molestie vel, gravida in, condime
    amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin vive
    odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisq
    </p>
  </div>
  <h3>部分 2</h3>
  <div>
    <p>
    Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum si
    purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turp
    velit, faucibus interdum tellus libero ac justo. Vivamus non qu
```

```
    suscipit faucibus urna.
    </p>
  </div>
  <h3>部分 3</h3>
  <div>
    <p>
    Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisc
    Phasellus pellentesque purus in massa. Aenean in pede. Phasellu
    ac tellus pellentesque semper. Sed ac felis. Sed commodo, magna
    lacinia ornare, quam ante aliquam nisi, eu iaculis leo purus ve
    </p>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
  <h3>部分 4</h3>
  <div>
    <p>
    Cras dictum. Pellentesque habitant morbi tristique senectus et
    et malesuada fames ac turpis egestas. Vestibulum ante ipsum pri
    faucibus orci luctus et ultrices posuere cubilia Curae; Aenean
    mauris vel est.
    </p>
    <p>
    Suspendisse eu nisl. Nullam ut libero. Integer dignissim consec
    Class aptent taciti sociosqu ad litora torquent per conubia nos
    inceptos himenaeos.
    </p>
  </div>
</div>

</body>
</html>
```

折叠内容

默认情况下，折叠面板总是保持一个部分是打开的。为了让所有部分都是折叠的，可设置 `collapsible` 选项为 true。点击当前打开的部分，来折叠它的内容面板。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 折叠内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      collapsible: true
    });
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget, q
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Qu
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
  <h3>部分 4</h3>
  <div>
    <p>Cras dictum. Pellentesque habitant morbi tristique senectus
  </div>
</div>

</body>
</html>
```

自定义图标

通过 `icons` 选项自定义标题图标，`icons` 选项接受标题默认的和激活的（打开的）状态的 class。使用 UI CSS 框架中的任意 class，或者使用背景图像创建自定义的 class。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 自定义图标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var icons = {
      header: "ui-icon-circle-arrow-e",
      activeHeader: "ui-icon-circle-arrow-s"
    };
    $( "#accordion" ).accordion({
      icons: icons
    });
    $( "#toggle" ).button().click(function() {
      if ( $( "#accordion" ).accordion( "option", "icons" ) ) {
        $( "#accordion" ).accordion( "option", "icons", null );
      } else {
        $( "#accordion" ).accordion( "option", "icons", icons );
      }
    });
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget, q
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Qu
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
```

```
    <h3>部分 4</h3>
    <div>
      <p>Cras dictum. Pellentesque habitant morbi tristique senectus
    </div>
  </div>

  <button id="toggle">切换图标</button>

  </body>
  </html>
```

填充空间

由于折叠面板是由块级元素组成的，默认情况下它的宽度会填充可用的水平空间。为了填充由容器分配的垂直空间，设置 `heightStyle` 选项为 `"fill"`，脚本会自动设置折叠面板的尺寸为父容器的高度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 填充空间</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #accordion-resizer {
    padding: 10px;
    width: 350px;
    height: 220px;
  }
  </style>
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      heightStyle: "fill"
    });
  });
  $(function() {
    $( "#accordion-resizer" ).resizable({
      minHeight: 140,
      minWidth: 200,
      resize: function() {
        $( "#accordion" ).accordion( "refresh" );
      }
    });
  });
  </script>
```

```
</head>
<body>

<h3 class="docs">重新调整外部容器：</h3>

<div id="accordion-resizer" class="ui-widget-content">
  <div id="accordion">
    <h3>部分 1</h3>
    <div>
      <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget,
    </div>
    <h3>部分 2</h3>
    <div>
      <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibul
    </div>
    <h3>部分 3</h3>
    <div>
      <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
      </ul>
    </div>
    <h3>部分 4</h3>
    <div>
      <p>Cras dictum. Pellentesque habitant morbi tristique senectu
    </div>
  </div>
</div>

</body>
</html>
```

非自动高度

设置 `heightStyle: "content"` ，让折叠面板保持它们初始的高度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 非自动高度</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      heightStyle: "content"
    });
  });
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, susceros. Nam mi
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Qu
    <ul>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
    </ul>
  </div>
</div>

</body>
</html>
```

当悬停时打开

点击头部展开/折叠被分为各个逻辑部分的内容，就像标签页（tabs）一样。您可以选择性地设置当鼠标悬停时是否切换各部分的打开/关闭状态。

基本的 HTML 标记是一系列的标题（H3 标签）和内容 div，因此内容不用通过 JavaScript 即可用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 当悬停时打开</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#accordion" ).accordion({
      event: "click hoverintent"
    });
  });

  /*
   * hoverIntent | Copyright 2011 Brian Cherne
   * http://cherne.net/brian/resources/jquery.hoverIntent.html
   * modified by the jQuery UI team
   */
  $.event.special.hoverintent = {
    setup: function() {
      $( this ).bind( "mouseover", jQuery.event.special.hoverintent
    },
    teardown: function() {
      $( this ).unbind( "mouseover", jQuery.event.special.hoverinte
    },
    handler: function( event ) {
      var currentX, currentY, timeout,
        args = arguments,
        target = $( event.target ),
        previousX = event.pageX,
        previousY = event.pageY;

      function track( event ) {
        currentX = event.pageX;
        currentY = event.pageY;
      };

      function clear() {
        target
          .unbind( "mousemove", track )
          .unbind( "mouseout", clear );
        clearTimeout( timeout );
      }
```

```
      function handler() {
        var prop,
          orig = event;

        if ( ( Math.abs( previousX - currentX ) +
            Math.abs( previousY - currentY ) ) < 7 ) {
          clear();

          event = $.Event( "hoverintent" );
          for ( prop in orig ) {
            if ( !( prop in event ) ) {
              event[ prop ] = orig[ prop ];
            }
          }
          // 防止访问原始事件，因为新事件会被异步触发，旧事件不再可用 (#6028
          delete event.originalEvent;

          target.trigger( event );
        } else {
          previousX = currentX;
          previousY = currentY;
          timeout = setTimeout( handler, 100 );
        }
      }

      timeout = setTimeout( handler, 100 );
      target.bind({
        mousemove: track,
        mouseout: clear
      });
    }
  };
  </script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>
    Mauris mauris ante, blandit et, ultrices a, suscipit eget, quar
    ut neque. Vivamus nisi metus, molestie vel, gravida in, condime
    amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin vive
    odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisc
    </p>
  </div>
  <h3>部分 2</h3>
  <div>
    <p>
    Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum si
    purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turp
    velit, faucibus interdum tellus libero ac justo. Vivamus non qu
```

```
      suscipit faucibus urna.
      </p>
    </div>
    <h3>部分 3</h3>
    <div>
      <p>
      Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisc
      Phasellus pellentesque purus in massa. Aenean in pede. Phasellu
      ac tellus pellentesque semper. Sed ac felis. Sed commodo, magna
      lacinia ornare, quam ante aliquam nisi, eu iaculis leo purus ve
      </p>
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
      </ul>
    </div>
    <h3>部分 4</h3>
    <div>
      <p>
      Cras dictum. Pellentesque habitant morbi tristique senectus et
      et malesuada fames ac turpis egestas. Vestibulum ante ipsum pri
      faucibus orci luctus et ultrices posuere cubilia Curae; Aenean
      mauris vel est.
      </p>
      <p>
      Suspendisse eu nisl. Nullam ut libero. Integer dignissim consec
      Class aptent taciti sociosqu ad litora torquent per conubia nos
      inceptos himenaeos.
      </p>
    </div>
  </div>

</body>
</html>
```

排序（Sortable）

拖拽标题来给面板重新排序。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 折叠面板（Accordion） - 排序（Sortable）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
```

```
  <style>
  /* 当排序时，IE 存在布局问题（查看 #5413） */
  .group { zoom: 1 }
  </style>
  <script>
  $(function() {
    $( "#accordion" )
      .accordion({
        header: "> div > h3"
      })
      .sortable({
        axis: "y",
        handle: "h3",
        stop: function( event, ui ) {
          // 当排序时，IE 不能注册 blur，所以触发 focusout 处理程序来移除
          ui.item.children( "h3" ).triggerHandler( "focusout" );
        }
      });
  });
  </script>
</head>
<body>

<div id="accordion">
  <div class="group">
    <h3>部分 1</h3>
    <div>
      <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget,
    </div>
  </div>
  <div class="group">
    <h3>部分 2</h3>
    <div>
      <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibul
    </div>
  </div>
  <div class="group">
    <h3>部分 3</h3>
    <div>
      <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
      <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
      </ul>
    </div>
  </div>
  <div class="group">
    <h3>部分 4</h3>
    <div>
      <p>Cras dictum. Pellentesque habitant morbi tristique senectu
    </div>
  </div>
```

```
</div>

</body>
</html>
```

jQuery UI 实例 - 自动完成（Autocomplete）

根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。

如需了解更多有关 autocomplete 部件的细节，请查看 API 文档 自动完成部件（Autocomplete Widget）。

默认功能

当您在输入域中输入时，自动完成（Autocomplete）部件提供相应的建议。在本实例中，提供了编程语言的建议选项，您可以输入 "ja" 尝试一下，可以得到 Java 或 JavaScript。

数据源是一个简单的 JavaScript 数组，使用 source 选项提供给部件。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
      "Scala",
      "Scheme"
    ];
    $( "#tags" ).autocomplete({
      source: availableTags
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">标签：</label>
  <input id="tags">
</div>

</body>
</html>
```

包含重音

autocomplete 域使用自定义的 source 选项来匹配带有重音字符的结果项，即使文本域不包含重音字符也会匹配。但是如果您在文本域中键入了重音字符，则不会显示非重音的结果项。

尝试键入 "Jo"，会看到 "John" 和 "J?rn"，然后 键入 "J?"，只会看到 "J?rn"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 包含重音</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var names = [ "J?rn Zaefferer", "Scott González", "John Resig"

    var accentMap = {
      "á": "a",
      "?": "o"
    };
    var normalize = function( term ) {
      var ret = "";
      for ( var i = 0; i < term.length; i++ ) {
        ret += accentMap[ term.charAt(i) ] || term.charAt(i);
      }
      return ret;
    };

    $( "#developer" ).autocomplete({
      source: function( request, response ) {
        var matcher = new RegExp( $.ui.autocomplete.escapeRegex( re
        response( $.grep( names, function( value ) {
          value = value.label || value.value || value;
          return matcher.test( value ) || matcher.test( normalize(
        }) );
      }
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <form>
  <label for="developer">开发人员：</label>
  <input id="developer">
  </form>
</div>

</body>
</html>
```

分类

分类的搜索结果。尝试键入 "a" 或 "n"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 分类</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-category {
    font-weight: bold;
    padding: .2em .4em;
    margin: .8em 0 .2em;
    line-height: 1.5;
  }
  </style>
  <script>
  $.widget( "custom.catcomplete", $.ui.autocomplete, {
    _renderMenu: function( ul, items ) {
      var that = this,
        currentCategory = "";
      $.each( items, function( index, item ) {
        if ( item.category != currentCategory ) {
          ul.append( "<li class='ui-autocomplete-category'>" + iter
          currentCategory = item.category;
        }
        that._renderItemData( ul, item );
      });
    }
  });
  </script>
  <script>
  $(function() {
    var data = [
      { label: "anders", category: "" },
      { label: "andreas", category: "" },
      { label: "antal", category: "" },
      { label: "annhhx10", category: "Products" },
      { label: "annk K12", category: "Products" },
      { label: "annttop C13", category: "Products" },
      { label: "anders andersson", category: "People" },
      { label: "andreas andersson", category: "People" },
      { label: "andreas johnson", category: "People" }
    ];

    $( "#search" ).catcomplete({
```

```
      delay: 0,
      source: data
    });
  });
  </script>
</head>
<body>

<label for="search">搜索：</label>
<input id="search">

</body>
</html>
```

组合框（Combobox）

一个由 Autocomplete 和 Button 创建的自定义部件。您可以键入一些字符，来获得基于您的输入过滤的结果，或者使用按钮从完整列表中选择。

该输入是从一个已有的 select 元素中读取，传递给带有自定义的 source 选项的 Autocomplete。

这是一个不被支持的不完美的部件。这里纯粹是为了演示 autocomplete 定制功能。如需了解更多有关该部件工作原理的细节，请点击这里查看相关的 jQuery 文章。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 组合框（Combobox）</titl
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .custom-combobox {
    position: relative;
    display: inline-block;
  }
  .custom-combobox-toggle {
    position: absolute;
    top: 0;
    bottom: 0;
    margin-left: -1px;
    padding: 0;
    /* 支持： IE7 */
    *height: 1.7em;
    *top: 0.1em;
```

```
}
.custom-combobox-input {
  margin: 0;
  padding: 0.3em;
}
</style>
<script>
(function( $ ) {
  $.widget( "custom.combobox", {
    _create: function() {
      this.wrapper = $( "<span>" )
        .addClass( "custom-combobox" )
        .insertAfter( this.element );

      this.element.hide();
      this._createAutocomplete();
      this._createShowAllButton();
    },

    _createAutocomplete: function() {
      var selected = this.element.children( ":selected" ),
        value = selected.val() ? selected.text() : "";

      this.input = $( "<input>" )
        .appendTo( this.wrapper )
        .val( value )
        .attr( "title", "" )
        .addClass( "custom-combobox-input ui-widget ui-widget-cor
        .autocomplete({
          delay: 0,
          minLength: 0,
          source: $.proxy( this, "_source" )
        })
        .tooltip({
          tooltipClass: "ui-state-highlight"
        });

      this._on( this.input, {
        autocompleteselect: function( event, ui ) {
          ui.item.option.selected = true;
          this._trigger( "select", event, {
            item: ui.item.option
          });
        },

        autocompletechange: "_removeIfInvalid"
      });
    },

    _createShowAllButton: function() {
      var input = this.input,
        wasOpen = false;
```

```
          $( "<a>" )
            .attr( "tabIndex", -1 )
            .attr( "title", "Show All Items" )
            .tooltip()
            .appendTo( this.wrapper )
            .button({
              icons: {
                primary: "ui-icon-triangle-1-s"
              },
              text: false
            })
            .removeClass( "ui-corner-all" )
            .addClass( "custom-combobox-toggle ui-corner-right" )
            .mousedown(function() {
              wasOpen = input.autocomplete( "widget" ).is( ":visible'
            })
            .click(function() {
              input.focus();

              // 如果已经可见则关闭
              if ( wasOpen ) {
                return;
              }

              // 传递空字符串作为搜索的值，显示所有的结果
              input.autocomplete( "search", "" );
            });
        },

        _source: function( request, response ) {
          var matcher = new RegExp( $.ui.autocomplete.escapeRegex(rec
          response( this.element.children( "option" ).map(function()
            var text = $( this ).text();
            if ( this.value && ( !request.term || matcher.test(text)
              return {
                label: text,
                value: text,
                option: this
              };
          }) );
        },

        _removeIfInvalid: function( event, ui ) {

          // 选择一项，不执行其他动作
          if ( ui.item ) {
            return;
          }

          // 搜索一个匹配（不区分大小写）
          var value = this.input.val(),
            valueLowerCase = value.toLowerCase(),
            valid = false;
```

```
        this.element.children( "option" ).each(function() {
          if ( $( this ).text().toLowerCase() === valueLowerCase )
            this.selected = valid = true;
            return false;
          }
        });

        // 找到一个匹配，不执行其他动作
        if ( valid ) {
          return;
        }

        // 移除无效的值
        this.input
          .val( "" )
          .attr( "title", value + " didn't match any item" )
          .tooltip( "open" );
        this.element.val( "" );
        this._delay(function() {
          this.input.tooltip( "close" ).attr( "title", "" );
        }, 2500 );
        this.input.data( "ui-autocomplete" ).term = "";
      },

      _destroy: function() {
        this.wrapper.remove();
        this.element.show();
      }
    });
  })( jQuery );

  $(function() {
    $( "#combobox" ).combobox();
    $( "#toggle" ).click(function() {
      $( "#combobox" ).toggle();
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label>您喜欢的编程语言：</label>
  <select id="combobox">
    <option value="">请选择...</option>
    <option value="ActionScript">ActionScript</option>
    <option value="AppleScript">AppleScript</option>
    <option value="Asp">Asp</option>
    <option value="BASIC">BASIC</option>
    <option value="C">C</option>
    <option value="C++">C++</option>
    <option value="Clojure">Clojure</option>
    <option value="COBOL">COBOL</option>
```

```
      <option value="ColdFusion">ColdFusion</option>
      <option value="Erlang">Erlang</option>
      <option value="Fortran">Fortran</option>
      <option value="Groovy">Groovy</option>
      <option value="Haskell">Haskell</option>
      <option value="Java">Java</option>
      <option value="JavaScript">JavaScript</option>
      <option value="Lisp">Lisp</option>
      <option value="Perl">Perl</option>
      <option value="PHP">PHP</option>
      <option value="Python">Python</option>
      <option value="Ruby">Ruby</option>
      <option value="Scala">Scala</option>
      <option value="Scheme">Scheme</option>
    </select>
  </div>
  <button id="toggle">显示基础的选择框</button>

  </body>
  </html>
```

自定义数据并显示

您可以使用自定义数据格式，并通过简单地重载默认的聚焦和选择行为来显示数据。

尝试键入 "j"，或者按向下箭头按键，即可得到一个项目列表。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 自定义数据并显示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #project-label {
    display: block;
    font-weight: bold;
    margin-bottom: 1em;
  }
  #project-icon {
    float: left;
    height: 32px;
    width: 32px;
  }
  #project-description {
```

```
    margin: 0;
    padding: 0;
  }
  </style>
  <script>
  $(function() {
    var projects = [
      {
        value: "jquery",
        label: "jQuery",
        desc: "the write less, do more, JavaScript library",
        icon: "jquery_32x32.png"
      },
      {
        value: "jquery-ui",
        label: "jQuery UI",
        desc: "the official user interface library for jQuery",
        icon: "jqueryui_32x32.png"
      },
      {
        value: "sizzlejs",
        label: "Sizzle JS",
        desc: "a pure-JavaScript CSS selector engine",
        icon: "sizzlejs_32x32.png"
      }
    ];

    $( "#project" ).autocomplete({
      minLength: 0,
      source: projects,
      focus: function( event, ui ) {
        $( "#project" ).val( ui.item.label );
        return false;
      },
      select: function( event, ui ) {
        $( "#project" ).val( ui.item.label );
        $( "#project-id" ).val( ui.item.value );
        $( "#project-description" ).html( ui.item.desc );
        $( "#project-icon" ).attr( "src", "images/" + ui.item.icon

        return false;
      }
    })
    .data( "ui-autocomplete" )._renderItem = function( ul, item ) {
      return $( "<li>" )
        .append( "<a>" + item.label + "<br>" + item.desc + "</a>" )
        .appendTo( ul );
    };
  });
  </script>
</head>
<body>
```

```
<div id="project-label">选择一个项目（请键入 "j"）：</div>
<img id="project-icon" src="images/transparent_1x1.png" class="ui-s
<input id="project">
<input type="hidden" id="project-id">
<p id="project-description"></p>

</body>
</html>
```

多个值

用法：键入一些字符，比如 "j"，可以看到相关的编程语言结果。选择一个值，然后继续键入字符来添加其他的值。

本实例演示如何使用 source 选项和一些事件来实现在一个单一的文本域输入多个自动完成的值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 多个值</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
```

```
      "Scala",
      "Scheme"
    ];
    function split( val ) {
      return val.split( /,\s*/ );
    }
    function extractLast( term ) {
      return split( term ).pop();
    }

    $( "#tags" )
      // 当选择一个条目时不离开文本域
      .bind( "keydown", function( event ) {
        if ( event.keyCode === $.ui.keyCode.TAB &&
            $( this ).data( "ui-autocomplete" ).menu.active ) {
          event.preventDefault();
        }
      })
      .autocomplete({
        minLength: 0,
        source: function( request, response ) {
          // 回到 autocomplete，但是提取最后的条目
          response( $.ui.autocomplete.filter(
            availableTags, extractLast( request.term ) ) );
        },
        focus: function() {
          // 防止在获得焦点时插入值
          return false;
        },
        select: function( event, ui ) {
          var terms = split( this.value );
          // 移除当前输入
          terms.pop();
          // 添加被选项
          terms.push( ui.item.value );
          // 添加占位符，在结尾添加逗号+空格
          terms.push( "" );
          this.value = terms.join( ", " );
          return false;
        }
      });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">编程语言：</label>
  <input id="tags" size="50">
</div>

</body>
</html>
```

多个值，远程

用法：键入至少两个字符来获取鸟的名称。选择一个值，然后继续键入字符来添加其他的值。

本实例演示如何使用 source 选项和一些事件来实现在一个单一的文本域输入多个自动完成的值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 多个值，远程</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  </style>
  <script>
  $(function() {
    function split( val ) {
      return val.split( /,\s*/ );
    }
    function extractLast( term ) {
      return split( term ).pop();
    }

    $( "#birds" )
      // 当选择一个条目时不离开文本域
      .bind( "keydown", function( event ) {
        if ( event.keyCode === $.ui.keyCode.TAB &&
            $( this ).data( "ui-autocomplete" ).menu.active ) {
          event.preventDefault();
        }
      })
      .autocomplete({
        source: function( request, response ) {
          $.getJSON( "search.php", {
            term: extractLast( request.term )
          }, response );
        },
        search: function() {
          // 自定义最小长度
          var term = extractLast( this.value );
```

```
            if ( term.length < 2 ) {
              return false;
            }
          },
          focus: function() {
            // 防止在获得焦点时插入值
            return false;
          },
          select: function( event, ui ) {
            var terms = split( this.value );
            // 移除当前输入
            terms.pop();
            // 添加被选项
            terms.push( ui.item.value );
            // 添加占位符，在结尾添加逗号+空格
            terms.push( "" );
            this.value = terms.join( ", " );
            return false;
          }
        });
    });
    </script>
  </head>
  <body>

  <div class="ui-widget">
    <label for="birds">鸟：</label>
    <input id="birds" size="50">
  </div>

  </body>
  </html>
```

远程 JSONP 数据源

当您在文本域中键入字符时，Autocomplete 部件给出建议结果。在本实例中，当您在文本域中至少键入两个字符时，将显示相关城市的名称。

在本实例中，数据源是 geonames.org webservice。虽然选择一个元素后文本域中是该城市名称，但是会显示更多的信息以便找到正确的条目。数据也可以回调，显示在下面的结果框中。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 远程 JSONP 数据源</title
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  #city { width: 25em; }
  </style>
  <script>
  $(function() {
    function log( message ) {
      $( "<div>" ).text( message ).prependTo( "#log" );
      $( "#log" ).scrollTop( 0 );
    }

    $( "#city" ).autocomplete({
      source: function( request, response ) {
        $.ajax({
          url: "http://ws.geonames.org/searchJSON",
          dataType: "jsonp",
          data: {
            featureClass: "P",
            style: "full",
            maxRows: 12,
            name_startsWith: request.term
          },
          success: function( data ) {
            response( $.map( data.geonames, function( item ) {
              return {
                label: item.name + (item.adminName1 ? ", " + item.a
                value: item.name
              }
            }));
          }
        });
      },
      minLength: 2,
      select: function( event, ui ) {
        log( ui.item ?
          "Selected: " + ui.item.label :
          "Nothing selected, input was " + this.value);
      },
      open: function() {
        $( this ).removeClass( "ui-corner-all" ).addClass( "ui-corr
      },
      close: function() {
        $( this ).removeClass( "ui-corner-top" ).addClass( "ui-corr
      }
    });
  });
  </script>
</head>
<body>
```

```
<div class="ui-widget">
  <label for="city">您的城市：</label>
  <input id="city">
  Powered by <a href="http://geonames.org" target="_blank">geonames
</div>

<div class="ui-widget" style="margin-top:2em; font-family:Arial">
  结果：
  <div id="log" style="height: 200px; width: 300px; overflow: auto;
</div>

</body>
</html>
```

远程数据源

当您在文本域中键入字符时，Autocomplete 部件给出建议结果。在本实例中，当您在文本域中至少键入两个字符时，将显示相关鸟的名称。

在本实例中，数据源是可返回 JSON 数据的服务器端脚本，通过一个简单的 source 选项来指定。另外，minLength 选项设置为 2，避免查询返回太多的结果，select 事件用于显示一些反馈。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 远程数据源</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  </style>
  <script>
  $(function() {
    function log( message ) {
      $( "<div>" ).text( message ).prependTo( "#log" );
      $( "#log" ).scrollTop( 0 );
    }

    $( "#birds" ).autocomplete({
      source: "search.php",
      minLength: 2,
      select: function( event, ui ) {
        log( ui.item ?
          "Selected: " + ui.item.value + " aka " + ui.item.id :
          "Nothing selected, input was " + this.value );
      }
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="birds">鸟：</label>
  <input id="birds">
</div>

<div class="ui-widget" style="margin-top:2em; font-family:Arial">
  结果：
  <div id="log" style="height: 200px; width: 300px; overflow: auto;
</div>

</body>
</html>
```

远程缓存

当您在文本域中键入字符时，Autocomplete 部件给出建议结果。在本实例中，当您在文本域中至少键入两个字符时，将显示相关鸟的名称。

为了提高性能，这里添加了一些本地缓存，其他与远程数据源实例相似。在这里，缓存只保存了一个查询，并可以扩展到缓存多个值，每个条目一个值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 远程缓存</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading {
    background: white url('images/ui-anim_basic_16x16.gif') right c
  }
  </style>
  <script>
  $(function() {
    var cache = {};
    $( "#birds" ).autocomplete({
      minLength: 2,
      source: function( request, response ) {
        var term = request.term;
        if ( term in cache ) {
          response( cache[ term ] );
          return;
        }

        $.getJSON( "search.php", request, function( data, status, x
          cache[ term ] = data;
          response( data );
        });
      }
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="birds">鸟：</label>
  <input id="birds">
</div>

</body>
</html>
```

可滚动的结果

当显示一个长列表的选项时，您可以简单地为 autocomplete 菜单设置 max-height 来防止菜单显示太长。尝试键入 "a" 或 "s" 来获得一个可滚动的长列表的结果。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - 可滚动的结果</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete {
    max-height: 100px;
    overflow-y: auto;
    /* 防止水平滚动条 */
    overflow-x: hidden;
  }
  /* IE 6 不支持 max-height
   * 我们使用 height 代替，但是这会强制菜单总是显示为那个高度
   */
  * html .ui-autocomplete {
    height: 100px;
  }
  </style>
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
```

```
        "Scala",
        "Scheme"
      ];
      $( "#tags" ).autocomplete({
        source: availableTags
      });
    });
    </script>
  </head>
  <body>

  <div class="ui-widget">
    <label for="tags">标签：</label>
    <input id="tags">
  </div>

  </body>
  </html>
```

XML 数据

本实例演示如何获取一些 XML 数据，并使用 jQuery 的方法解析它，然后把它提供给 autocomplete 作为数据源。

本实例也可作为解析远程 XML 数据源的参考 - 解析在每次 source 回调请求时发生。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 自动完成（Autocomplete） - XML 数据</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-autocomplete-loading { background: white url('images/ui-anim_
  </style>
  <script>
  $(function() {
    function log( message ) {
      $( "<div/>" ).text( message ).prependTo( "#log" );
      $( "#log" ).attr( "scrollTop", 0 );
    }

    $.ajax({
      url: "london.xml",
      dataType: "xml",
```

```
        success: function( xmlResponse ) {
          var data = $( "geoname", xmlResponse ).map(function() {
            return {
              value: $( "name", this ).text() + ", " +
                ( $.trim( $( "countryName", this ).text() ) || "(unkn
              id: $( "geonameId", this ).text()
            };
          }).get();
          $( "#birds" ).autocomplete({
            source: data,
            minLength: 0,
            select: function( event, ui ) {
              log( ui.item ?
                "Selected: " + ui.item.value + ", geonameId: " + ui.i
                "Nothing selected, input was " + this.value );
            }
          });
        }
      });
    });
    </script>
  </head>
  <body>

  <div class="ui-widget">
    <label for="birds">London 匹配：</label>
    <input id="birds">
  </div>

  <div class="ui-widget" style="margin-top:2em; font-family:Arial">
    结果：
    <div id="log" style="height: 200px; width: 300px; overflow: auto;
  </div>

  </body>
  </html>
```

jQuery UI 实例 - 按钮（Button）

用带有适当的悬停（hover）和激活（active）的样式的可主题化按钮来加强标准表单元素（比如按钮、输入框、锚）的功能。

如需了解更多有关 button 部件的细节，请查看 API 文档 按钮部件（Button Widget）。

默认功能

可用于按钮的标记实例：一个 button 元素，一个类型为 submit 的 input 元素和一个锚。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "input[type=submit], a, button" )
      .button()
      .click(function( event ) {
        event.preventDefault();
      });
  });
  </script>
</head>
<body>

<button>一个 button 元素</button>

<input type="submit" value="一个提交按钮">

<a href="#">一个锚</a>

</body>
</html>
```

复选框

通过 button 部件把复选框显示为切换按钮样式。与复选框相关的 label 元素作为按钮文本。

本实例通过在一个公共的容器上调用 `.buttonset()` ，演示了三个显示为按钮样式的复选框。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 复选框</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#check" ).button();
    $( "#format" ).buttonset();
  });
  </script>
  <style>
  #format { margin-top: 2em; }
  </style>
</head>
<body>

<input type="checkbox" id="check"><label for="check">切换</label>

<div id="format">
  <input type="checkbox" id="check1"><label for="check1">B</label>
  <input type="checkbox" id="check2"><label for="check2">I</label>
  <input type="checkbox" id="check3"><label for="check3">U</label>
</div>

</body>
</html>
```

图标

一些带有文本和图标的各种组合的按钮

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 图标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "button:first" ).button({
      icons: {
        primary: "ui-icon-locked"
      },
      text: false
    }).next().button({
      icons: {
        primary: "ui-icon-locked"
      }
    }).next().button({
      icons: {
        primary: "ui-icon-gear",
        secondary: "ui-icon-triangle-1-s"
      }
    }).next().button({
      icons: {
        primary: "ui-icon-gear",
        secondary: "ui-icon-triangle-1-s"
      },
      text: false
    });
  });
  </script>
</head>
<body>

<button>只带有图标的按钮</button>
<button>图标在左侧的按钮</button>
<button>带有两个图标的按钮</button>
<button>带有两个图标且不带文本的按钮</button>

</body>
</html>
```

单选

三个单选按钮转变为一套按钮。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 单选</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#radio" ).buttonset();
  });
  </script>
</head>
<body>

<form>
  <div id="radio">
    <input type="radio" id="radio1" name="radio"><label for="radio1
    <input type="radio" id="radio2" name="radio" checked="checked">
    <input type="radio" id="radio3" name="radio"><label for="radio3
  </div>
</form>

</body>
</html>
```

分割按钮

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 分割按钮</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .ui-menu { position: absolute; width: 100px; }
  </style>
  <script>
  $(function() {
    $( "#rerun" )
      .button()
      .click(function() {
        alert( "Running the last action" );
```

```
        })
        .next()
          .button({
            text: false,
            icons: {
              primary: "ui-icon-triangle-1-s"
            }
          })
          .click(function() {
            var menu = $( this ).parent().next().show().position({
              my: "left top",
              at: "left bottom",
              of: this
            });
            $( document ).one( "click", function() {
              menu.hide();
            });
            return false;
          })
          .parent()
            .buttonset()
            .next()
              .hide()
              .menu();
  });
  </script>
</head>
<body>

<div>
  <div>
    <button id="rerun">运行最后的动作</button>
    <button id="select">选择一个动作</button>
  </div>
  <ul>
    <li><a href="#">打开...</a></li>
    <li><a href="#">保存</a></li>
    <li><a href="#">删除</a></li>
  </ul>
</div>

</body>
</html>
```

工具栏

一个多媒体播放器的工具栏。请看基础的标记：一些 button 元素，Shuffle 按钮是一个类型为 checkbox 的 input，Repeat 选项是三个类型为 radio 的 input。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 按钮（Button） - 工具栏</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #toolbar {
    padding: 4px;
    display: inline-block;
  }
  /* support: IE7 */
  *+html #toolbar {
    display: inline;
  }
  </style>
  <script>
  $(function() {
    $( "#beginning" ).button({
      text: false,
      icons: {
        primary: "ui-icon-seek-start"
      }
    });
    $( "#rewind" ).button({
      text: false,
      icons: {
        primary: "ui-icon-seek-prev"
      }
    });
    $( "#play" ).button({
      text: false,
      icons: {
        primary: "ui-icon-play"
      }
    })
    .click(function() {
      var options;
      if ( $( this ).text() === "play" ) {
        options = {
          label: "pause",
          icons: {
            primary: "ui-icon-pause"
          }
        };
      } else {
        options = {
          label: "play",
          icons: {
```

```
              primary: "ui-icon-play"
            }
          };
        }
        $( this ).button( "option", options );
      });
      $( "#stop" ).button({
        text: false,
        icons: {
          primary: "ui-icon-stop"
        }
      })
      .click(function() {
        $( "#play" ).button( "option", {
          label: "play",
          icons: {
            primary: "ui-icon-play"
          }
        });
      });
      $( "#forward" ).button({
        text: false,
        icons: {
          primary: "ui-icon-seek-next"
        }
      });
      $( "#end" ).button({
        text: false,
        icons: {
          primary: "ui-icon-seek-end"
        }
      });
      $( "#shuffle" ).button();
      $( "#repeat" ).buttonset();
    });
    </script>
  </head>
  <body>

  <div id="toolbar" class="ui-widget-header ui-corner-all">
    <button id="beginning">开头</button>
    <button id="rewind">快退</button>
    <button id="play">播放</button>
    <button id="stop">停止</button>
    <button id="forward">快进</button>
    <button id="end">结尾</button>

    <input type="checkbox" id="shuffle"><label for="shuffle">Shuffle<

    <span id="repeat">
      <input type="radio" id="repeat0" name="repeat" checked="checked
      <input type="radio" id="repeat1" name="repeat"><label for="repe
      <input type="radio" id="repeatall" name="repeat"><label for="re
```

```
  </span>
</div>

</body>
</html>
```

'Attr.nodeValue' is deprecated. Please use 'value' instead. adsbygoogle.js:32 href http://www.w3cschool.cc/jqueryui/example-datepicker.html example-datepicker.html:438 tmp_url http://www.w3cschool.cc/jqueryui/example-datepicker.html example-datepicker.html:450 example-button.html jQuery UI 实例 – 按钮（Button） example-datepicker.html:495 example-dialog.html jQuery UI 实例 – 对话框（Dialog） example-datepicker.html:496

jQuery UI 实例 - 日期选择器（Datepicker）

从弹出框或内联日历选择一个日期。

如需了解更多有关 datepicker 部件的细节，请查看 API 文档 日期选择器部件（Datepicker Widget）。

默认功能

日期选择器（Datepicker）绑定到一个标准的表单 input 字段上。把焦点移到 input 上（点击或者使用 tab 键），在一个小的覆盖层上打开一个交互日历。选择一个日期，点击页面上的任意地方（输入框即失去焦点），或者点击 Esc 键来关闭。如果选择了一个日期，则反馈显示为 input 的值。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker();
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

动画

当打开或关闭 datepicker 时使用不同的动画。从下拉框中选择一个动画，然后在输入框中点击来查看它的效果。您可以使用三个标准动画中任意一个，或者使用 UI 特效中的任意一个。

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>jQuery UI 日期选择器（Datepicker） - 动画</title>
    <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes
    <script src="//code.jquery.com/jquery-1.9.1.js"></script>
    <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script
    <link rel="stylesheet" href="http://jqueryui.com/resources/demo
    <script>
    $(function() {
        $( "#datepicker" ).datepicker();
        $( "#anim" ).change(function() {
            $( "#datepicker" ).datepicker( "option", "showAnim", $(
        });
    });
    </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker" size="30"></p>

<p>动画：<br>
    <select id="anim">
        <option value="show">Show (默认)</option>
        <option value="slideDown">滑下</option>
        <option value="fadeIn">淡入</option>
        <option value="blind">Blind (UI 百叶窗特效)</option>
        <option value="bounce">Bounce (UI 反弹特效)</option>
        <option value="clip">Clip (UI 剪辑特效)</option>
        <option value="drop">Drop (UI 降落特效)</option>
        <option value="fold">Fold (UI 折叠特效)</option>
        <option value="slide">Slide (UI 滑动特效)</option>
        <option value="">无</option>
    </select>
</p>

</body>
</html>
```

其他月份的日期

datepicker 可以显示其他月份的日期，这些日期也可以设置成可选择的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 其他月份的日期</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showOtherMonths: true,
      selectOtherMonths: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

显示按钮栏

通过布尔值的 `showButtonPanel` 选项为选择当天日期显示一个"Today"按钮，为关闭日历显示一个"Done"按钮。默认情况下，当按钮栏显示时会启用每个按钮，但是按钮可通过其他的选项进行关闭。按钮文本可自定义。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示按钮栏</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showButtonPanel: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

内联显示

datepicker 是嵌套在页面中显示，而不是显示在一个覆盖层中。只需要简单地在 div 上调用 .datepicker() 即可，而不是在 input 上调用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 内联显示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker();
  });
  </script>
</head>
<body>

日期：<div id="datepicker"></div>

</body>
</html>
```

显示月份 & 年份菜单

显示月份和年份的下拉框，而不是显示静态的月份/年份标题，这样便于在大范围的时间跨度上导航。添加布尔值 `changeMonth` 和 `changeYear` 选项即可。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示月份 &amp; 年份菜单</t
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      changeMonth: true,
      changeYear: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

显示多个月份

设置 `numberOfMonths` 选项为一个整数 2，或者大于 2 的整数，来在一个 datepicker 中显示多个月份。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示多个月份</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      numberOfMonths: 3,
      showButtonPanel: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

格式化日期

以各种方式显示日期反馈。从下拉框中选择一种日期格式，然后在输入框中点击并选择一个日期，查看所选格式的日期显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 格式化日期</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker();
    $( "#format" ).change(function() {
      $( "#datepicker" ).datepicker( "option", "dateFormat", $( thi
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker" size="30"></p>

<p>格式选项：<br>
  <select id="format">
    <option value="mm/dd/yy">Default - mm/dd/yy</option>
    <option value="yy-mm-dd">ISO 8601 - yy-mm-dd</option>
    <option value="d M, y">Short - d M, y</option>
    <option value="d MM, y">Medium - d MM, y</option>
    <option value="DD, d MM, yy">Full - DD, d MM, yy</option>
    <option value="&apos;day&apos; d &apos;of&apos; MM &apos;in the
  </select>
</p>

</body>
</html>
```

图标触发器

点击输入框旁边的图标来显示 datepicker。设置 datepicker 在获得焦点时打开（默认行为），或者在点击图标时打开，或者在获得焦点/点击图标时打开。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 图标触发器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showOn: "button",
      buttonImage: "images/calendar.gif",
      buttonImageOnly: true
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

本地化日历

本地化 datepicker 日历语言和格式（默认为 English / Western 格式）。datepicker 包含对从右到左读取的语言的内建支持，比如 Arabic 和 Hebrew。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 本地化日历</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <script src="http://jqueryui.com/resources/demos/datepicker/jquer
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker( $.datepicker.regional[ "fr" ] );
    $( "#locale" ).change(function() {
      $( "#datepicker" ).datepicker( "option",
        $.datepicker.regional[ $( this ).val() ] );
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"> 
  <select id="locale">
    <option value="ar">Arabic (&#8235;(&#1575;&#1604;&#1593;&#1585;
    <option value="zh-TW">Chinese Traditional (&#32321;&#39636;&#20
    <option value="">English</option>
    <option value="fr" selected="selected">French (Fran&ccedil;ais)
    <option value="he">Hebrew (&#8235;(&#1506;&#1489;&#1512;&#1497;
  </select></p>

</body>
</html>
```

填充另一个输入框

使用 `altField` 和 `altFormat` 选项，无论何时选择日期，会在另一个输入框中填充带有一定格式的日期。这个功能通过对电脑友好性的日期进一步加工后，向用户呈现一个用户友好性的日期。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 填充另一个输入框</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      altField: "#alternate",
      altFormat: "DD, d MM, yy"
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"> <input type="text"

</body>
</html>
```

限制日期范围

通过 `minDate` 和 `maxDate` 选项限制可选择的日期范围。设置起止日期为实际的日期（new Date(2009, 1 - 1, 26)），或者为与今天的一个数值偏移（-20），或者为一个周期和单位的字符串（'+1M +10D'）。如果设置为字符串，使用 'D' 表示天，使用 'W' 表示周，使用 'M' 表示月，使用 'Y' 表示年。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 限制日期范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({ minDate: -20, maxDate: "+1M +10
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

选择一个日期范围

选择要搜索的日期范围。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 选择一个日期范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#from" ).datepicker({
      defaultDate: "+1w",
      changeMonth: true,
      numberOfMonths: 3,
      onClose: function( selectedDate ) {
        $( "#to" ).datepicker( "option", "minDate", selectedDate );
      }
    });
    $( "#to" ).datepicker({
      defaultDate: "+1w",
      changeMonth: true,
      numberOfMonths: 3,
      onClose: function( selectedDate ) {
        $( "#from" ).datepicker( "option", "maxDate", selectedDate
      }
    });
  });
  </script>
</head>
<body>

<label for="from">从</label>
<input type="text" id="from" name="from">
<label for="to">到</label>
<input type="text" id="to" name="to">

</body>
</html>
```

显示一年中的第几周

datepicker 可以显示一年中的第几周。默认的计算是按照 ISO 8601 定义：每周从星期一开始，每年的第一周包含该年的第一个星期四。这就意味着一年中的一些天可能是属于另一年中的周。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 日期选择器（Datepicker） - 显示一年中的第几周</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#datepicker" ).datepicker({
      showWeek: true,
      firstDay: 1
    });
  });
  </script>
</head>
<body>

<p>日期：<input type="text" id="datepicker"></p>

</body>
</html>
```

jQuery UI 实例 - 对话框（Dialog）

在一个交互覆盖层中打开内容。

如需了解更多有关 dialog 部件的细节，请查看 API 文档 对话框部件（Dialog Widget）。

默认功能

基本的对话框窗口是一个定位于视区中的覆盖层，同时通过一个 iframe 与页面内容分隔开（就像 select 元素）。它由一个标题栏和一个内容区域组成，且可以移动，调整尺寸，默认可通过 'x' 图标关闭。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog" ).dialog();
  });
  </script>
</head>
<body>

<div id="dialog" title="基本的对话框">
  <p>这是一个默认的对话框，用于显示信息。对话框窗口可以移动，调整尺寸，默认可通
</div>

</body>
</html>
```

动画

可以通过为 show/hide 属性指定一个特效来动画显示对话框。您必须为想使用的特效引用独立的特效文件。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 动画</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog" ).dialog({
      autoOpen: false,
      show: {
        effect: "blind",
        duration: 1000
      },
      hide: {
        effect: "explode",
        duration: 1000
      }
    });

    $( "#opener" ).click(function() {
      $( "#dialog" ).dialog( "open" );
    });
  });
  </script>
</head>
<body>

<div id="dialog" title="Basic dialog">
  <p>这是一个动画显示的对话框，用于显示信息。对话框窗口可以移动，调整尺寸，默认
</div>

<button id="opener">打开对话框</button>

</body>
</html>
```

基本的模态

模态对话框防止用户与对话框以外的页面其他部分进行交互，直到对话框关闭为止。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 基本的模态</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog-modal" ).dialog({
      height: 140,
      modal: true
    });
  });
  </script>
</head>
<body>

<div id="dialog-modal" title="基本的模态对话框">
  <p>添加模态覆盖屏幕，让对话框看起来更突出，因为它让页面上其他内容变暗。</p>
</div>

<p>Sed vel diam id libero <a href="http://www.w3cschool.cc">rutrum

</body>
</html>
```

模态确认

确认一个动作可能是破坏性的也可能是有意义的。设置 `modal` 选项为 true，并通过 `buttons` 选项来指定主要的和次要的用户动作。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 模态确认</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog-confirm" ).dialog({
      resizable: false,
      height:140,
      modal: true,
      buttons: {
        "Delete all items": function() {
          $( this ).dialog( "close" );
        },
        Cancel: function() {
          $( this ).dialog( "close" );
        }
      }
    });
  });
  </script>
</head>
<body>

<div id="dialog-confirm" title="清空回收站吗？">
  <p><span class="ui-icon ui-icon-alert" style="float:left; margin
</div>

<p>Sed vel diam id libero <a href="http://www.w3cschool.cc">rutrum

</body>
</html>
```

模态表单

使用模态对话框来请求用户在一个多步骤过程中输入数据。在内容区域嵌入 form 标记，设置 `modal` 选项为 true，并通过 `buttons` 选项来指定主要的和次要的用户动作。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
  <title>jQuery UI 对话框（Dialog） - 模态表单</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    body { font-size: 62.5%; }
    label, input { display:block; }
    input.text { margin-bottom:12px; width:95%; padding: .4em; }
    fieldset { padding:0; border:0; margin-top:25px; }
    h1 { font-size: 1.2em; margin: .6em 0; }
    div#users-contain { width: 350px; margin: 20px 0; }
    div#users-contain table { margin: 1em 0; border-collapse: colla
    div#users-contain table td, div#users-contain table th { border
    .ui-dialog .ui-state-error { padding: .3em; }
    .validateTips { border: 1px solid transparent; padding: 0.3em;
  </style>
  <script>
  $(function() {
    var name = $( "#name" ),
      email = $( "#email" ),
      password = $( "#password" ),
      allFields = $( [] ).add( name ).add( email ).add( password ),
      tips = $( ".validateTips" );

    function updateTips( t ) {
      tips
        .text( t )
        .addClass( "ui-state-highlight" );
      setTimeout(function() {
        tips.removeClass( "ui-state-highlight", 1500 );
      }, 500 );
    }

    function checkLength( o, n, min, max ) {
      if ( o.val().length > max || o.val().length < min ) {
        o.addClass( "ui-state-error" );
        updateTips( "" + n + " 的长度必须在 " +
          min + " 和 " + max + " 之间。" );
        return false;
      } else {
        return true;
      }
    }

    function checkRegexp( o, regexp, n ) {
      if ( !( regexp.test( o.val() ) ) ) {
        o.addClass( "ui-state-error" );
        updateTips( n );
        return false;
      } else {
        return true;
      }
```

```
    }

    $( "#dialog-form" ).dialog({
      autoOpen: false,
      height: 300,
      width: 350,
      modal: true,
      buttons: {
        "创建一个帐户": function() {
          var bValid = true;
          allFields.removeClass( "ui-state-error" );

          bValid = bValid && checkLength( name, "username", 3, 16 )
          bValid = bValid && checkLength( email, "email", 6, 80 );
          bValid = bValid && checkLength( password, "password", 5,

          bValid = bValid && checkRegexp( name, /^[a-z]([0-9a-z_])-
          // From jquery.validate.js (by joern), contributed by Sco
          bValid = bValid && checkRegexp( email, /^((([a-z]|\d|[!#
          bValid = bValid && checkRegexp( password, /^([0-9a-zA-Z]

          if ( bValid ) {
            $( "#users tbody" ).append( "<tr>" +
              "<td>" + name.val() + "</td>" +
              "<td>" + email.val() + "</td>" +
              "<td>" + password.val() + "</td>" +
            "</tr>" );
            $( this ).dialog( "close" );
          }
        },
        Cancel: function() {
          $( this ).dialog( "close" );
        }
      },
      close: function() {
        allFields.val( "" ).removeClass( "ui-state-error" );
      }
    });

    $( "#create-user" )
      .button()
      .click(function() {
        $( "#dialog-form" ).dialog( "open" );
      });
  });
  </script>
</head>
<body>

<div id="dialog-form" title="创建新用户">
  <p class="validateTips">所有的表单字段都是必填的。</p>

  <form>
```

```
    <fieldset>
      <label for="name">名字</label>
      <input type="text" name="name" id="name" class="text ui-widget-
      <label for="email">邮箱</label>
      <input type="text" name="email" id="email" value="" class="text
      <label for="password">密码</label>
      <input type="password" name="password" id="password" value="" c
    </fieldset>
    </form>
  </div>

  <div id="users-contain" class="ui-widget">
    <h1>已有的用户：</h1>
    <table id="users" class="ui-widget ui-widget-content">
      <thead>
        <tr class="ui-widget-header ">
          <th>名字</th>
          <th>邮箱</th>
          <th>密码</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>John Doe</td>
          <td>john.doe@example.com</td>
          <td>johndoe1</td>
        </tr>
      </tbody>
    </table>
  </div>
  <button id="create-user">创建新用户</button>

  </body>
  </html>
```

模态消息

使用模态对话框来在下一步动作执行之前确认信息和动作。设置 `modal` 选项为 true，并通过 `buttons` 选项来指定主要的动作（Ok）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 对话框（Dialog） - 模态消息</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#dialog-message" ).dialog({
      modal: true,
      buttons: {
        Ok: function() {
          $( this ).dialog( "close" );
        }
      }
    });
  });
  </script>
</head>
<body>

<div id="dialog-message" title="下载完成">
  <p>
    <span class="ui-icon ui-icon-circle-check" style="float:left; r
    您的文件已经成功下载到文件夹中。
  </p>
  <p>
    当前使用存储空间的 <b>36%</b>。
  </p>
</div>

<p>Sed vel diam id libero <a href="http://www.w3cschool.cc">rutrum

</body>
</html>
```

jQuery UI 实例 - 菜单（Menu）

带有鼠标和键盘交互的用于导航的可主题化菜单。

如需了解更多有关 menu 部件的细节，请查看 API 文档 菜单部件（Menu Widget）。

默认功能

一个带有默认配置、禁用条目和嵌套菜单的菜单。由一个列表转化成的，添加了主题，并支持鼠标和键盘交互。尝试使用光标键导航菜单。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 菜单（Menu） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#menu" ).menu();
  });
  </script>
  <style>
  .ui-menu { width: 150px; }
  </style>
</head>
<body>

<ul id="menu">
  <li class="ui-state-disabled"><a href="#">Aberdeen</a></li>
  <li><a href="#">Ada</a></li>
  <li><a href="#">Adamsville</a></li>
  <li><a href="#">Addyston</a></li>
  <li>
    <a href="#">Delphi</a>
    <ul>
      <li class="ui-state-disabled"><a href="#">Ada</a></li>
      <li><a href="#">Saarland</a></li>
      <li><a href="#">Salzburg</a></li>
    </ul>
  </li>
  <li><a href="#">Saarland</a></li>
  <li>
    <a href="#">Salzburg</a>
```

```
      <ul>
        <li>
          <a href="#">Delphi</a>
          <ul>
            <li><a href="#">Ada</a></li>
            <li><a href="#">Saarland</a></li>
            <li><a href="#">Salzburg</a></li>
          </ul>
        </li>
        <li>
          <a href="#">Delphi</a>
          <ul>
            <li><a href="#">Ada</a></li>
            <li><a href="#">Saarland</a></li>
            <li><a href="#">Salzburg</a></li>
          </ul>
        </li>
        <li><a href="#">Perch</a></li>
      </ul>
    </li>
    <li class="ui-state-disabled"><a href="#">Amesville</a></li>
  </ul>

</body>
</html>
```

图标

一个带有默认配置的菜单，显示如何使用带有图标的菜单。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 菜单（Menu） - 图标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#menu" ).menu();
  });
  </script>
  <style>
  .ui-menu { width: 150px; }
  </style>
</head>
<body>

<ul id="menu">
  <li><a href="#"><span class="ui-icon ui-icon-disk"></span>保存</a
  <li><a href="#"><span class="ui-icon ui-icon-zoomin"></span>放大<
  <li><a href="#"><span class="ui-icon ui-icon-zoomout"></span>缩小
  <li class="ui-state-disabled"><a href="#"><span class="ui-icon ui
  <li>
    <a href="#">播放</a>
    <ul>
      <li><a href="#"><span class="ui-icon ui-icon-seek-start"></sp
      <li><a href="#"><span class="ui-icon ui-icon-stop"></span>停止
      <li><a href="#"><span class="ui-icon ui-icon-play"></span>播放
      <li><a href="#"><span class="ui-icon ui-icon-seek-end"></span
    </ul>
  </li>
</ul>

</body>
</html>
```

jQuery UI 实例 - 进度条（Progressbar）

显示一个确定的或不确定的进程状态。

如需了解更多有关 progressbar 部件的细节，请查看 API 文档 进度条部件（Progressbar Widget）。

默认功能

默认的确定的进度条。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 进度条（Progressbar） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#progressbar" ).progressbar({
      value: 37
    });
  });
  </script>
</head>
<body>

<div id="progressbar"></div>

</body>
</html>
```

自定义标签

自定义更新的标签。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 进度条（Progressbar） - 自定义标签</title>
```

```
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .ui-progressbar {
    position: relative;
  }
  .progress-label {
    position: absolute;
    left: 50%;
    top: 4px;
    font-weight: bold;
    text-shadow: 1px 1px 0 #fff;
  }
  </style>
  <script>
  $(function() {
    var progressbar = $( "#progressbar" ),
      progressLabel = $( ".progress-label" );

    progressbar.progressbar({
      value: false,
      change: function() {
        progressLabel.text( progressbar.progressbar( "value" ) + "%
      },
      complete: function() {
        progressLabel.text( "完成！" );
      }
    });

    function progress() {
      var val = progressbar.progressbar( "value" ) || 0;

      progressbar.progressbar( "value", val + 1 );

      if ( val < 99 ) {
        setTimeout( progress, 100 );
      }
    }

    setTimeout( progress, 3000 );
  });
  </script>
</head>
<body>

<div id="progressbar"><div class="progress-label">加载...</div></di

</body>
</html>
```

不确定的值

不确定的进度条，并可在确定和不确定的样式之间切换。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 进度条（Progressbar） - 不确定的值</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#progressbar" ).progressbar({
      value: false
    });
    $( "button" ).on( "click", function( event ) {
      var target = $( event.target ),
        progressbar = $( "#progressbar" ),
        progressbarValue = progressbar.find( ".ui-progressbar-value

      if ( target.is( "#numButton" ) ) {
        progressbar.progressbar( "option", {
          value: Math.floor( Math.random() * 100 )
        });
      } else if ( target.is( "#colorButton" ) ) {
        progressbarValue.css({
          "background": '#' + Math.floor( Math.random() * 16777215
        });
      } else if ( target.is( "#falseButton" ) ) {
        progressbar.progressbar( "option", "value", false );
      }
    });
  });
  </script>
  <style>
  #progressbar .ui-progressbar-value {
    background-color: #ccc;
  }
  </style>
</head>
<body>

<div id="progressbar"></div>
<button id="numButton">随机值 - 确定</button>
<button id="falseButton">不确定</button>
<button id="colorButton">随机颜色</button>

</body>
</html>
```

jQuery UI 实例 - 滑块（Slider）

拖动手柄来选择一个数值。

如需了解更多有关 slider 部件的细节，请查看 API 文档 滑块部件（Slider Widget）。

默认功能

基本的滑块是水平的，有一个单一的手柄，可以用鼠标或箭头键进行移动。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider" ).slider();
  });
  </script>
</head>
<body>

<div id="slider"></div>

</body>
</html>
```

颜色选择器

组合了三个滑块，来创建一个简单的 RGB 颜色选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 颜色选择器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
```

```
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #red, #green, #blue {
    float: left;
    clear: left;
    width: 300px;
    margin: 15px;
  }
  #swatch {
    width: 120px;
    height: 100px;
    margin-top: 18px;
    margin-left: 350px;
    background-image: none;
  }
  #red .ui-slider-range { background: #ef2929; }
  #red .ui-slider-handle { border-color: #ef2929; }
  #green .ui-slider-range { background: #8ae234; }
  #green .ui-slider-handle { border-color: #8ae234; }
  #blue .ui-slider-range { background: #729fcf; }
  #blue .ui-slider-handle { border-color: #729fcf; }
  </style>
  <script>
  function hexFromRGB(r, g, b) {
    var hex = [
      r.toString( 16 ),
      g.toString( 16 ),
      b.toString( 16 )
    ];
    $.each( hex, function( nr, val ) {
      if ( val.length === 1 ) {
        hex[ nr ] = "0" + val;
      }
    });
    return hex.join( "" ).toUpperCase();
  }
  function refreshSwatch() {
    var red = $( "#red" ).slider( "value" ),
      green = $( "#green" ).slider( "value" ),
      blue = $( "#blue" ).slider( "value" ),
      hex = hexFromRGB( red, green, blue );
    $( "#swatch" ).css( "background-color", "#" + hex );
  }
  $(function() {
    $( "#red, #green, #blue" ).slider({
      orientation: "horizontal",
      range: "min",
      max: 255,
      value: 127,
      slide: refreshSwatch,
      change: refreshSwatch
    });
```

```
    $( "#red" ).slider( "value", 255 );
    $( "#green" ).slider( "value", 140 );
    $( "#blue" ).slider( "value", 60 );
  });
  </script>
</head>
<body class="ui-widget-content" style="border:0;">

<p class="ui-state-default ui-corner-all ui-helper-clearfix" style=
  <span class="ui-icon ui-icon-pencil" style="float:left; margin:-2
  简单的颜色选择器
</p>

<div id="red"></div>
<div id="green"></div>
<div id="blue"></div>

<div id="swatch" class="ui-widget-content ui-corner-all"></div>

</body>
</html>
```

多个滑块

组合水平的和垂直的滑块，每个都带有各自的选项，来创建一个音乐播放器的 UI。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 多个滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #eq span {
    height:120px; float:left; margin:15px
  }
  </style>
  <script>
  $(function() {
    // 设置主音量
    $( "#master" ).slider({
      value: 60,
      orientation: "horizontal",
      range: "min",
      animate: true
    });
```

```
    // 设置图形均衡器
    $( "#eq > span" ).each(function() {
      // 从标记读取初始值并删除
      var value = parseInt( $( this ).text(), 10 );
      $( this ).empty().slider({
        value: value,
        range: "min",
        animate: true,
        orientation: "vertical"
      });
    });
  });
  </script>
</head>
<body>

<p class="ui-state-default ui-corner-all ui-helper-clearfix" style=
  <span class="ui-icon ui-icon-volume-on" style="float:left; margin
  主音量
</p>

<div id="master" style="width:260px; margin:15px;"></div>

<p class="ui-state-default ui-corner-all" style="padding:4px;margin
  <span class="ui-icon ui-icon-signal" style="float:left; margin:-2
  图形均衡器
</p>

<div id="eq">
  <span>88</span>
  <span>77</span>
  <span>55</span>
  <span>33</span>
  <span>40</span>
  <span>45</span>
  <span>70</span>
</div>

</body>
</html>
```

范围滑块

设置 `range` 选项为 true，来获取带有两个拖拽手柄的值的范围。手柄之间的控件用不同的背景颜色填充来表示该区间的值是可选择的。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 范围滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range" ).slider({
      range: true,
      min: 0,
      max: 500,
      values: [ 75, 300 ],
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.values[ 0 ] + " - $" + ui.valu
      }
    });
    $( "#amount" ).val( "$" + $( "#slider-range" ).slider( "values"
      " - $" + $( "#slider-range" ).slider( "values", 1 ) );
  });
  </script>
</head>
<body>

<p>
  <label for="amount">价格范围：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-range"></div>

</body>
</html>
```

带有固定最大值的范围

固定范围滑块的最大值，用户只能选择最小值。设置 `range` 选项为 "max"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 带有固定最大值的范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range-max" ).slider({
      range: "max",
      min: 1,
      max: 10,
      value: 2,
      slide: function( event, ui ) {
        $( "#amount" ).val( ui.value );
      }
    });
    $( "#amount" ).val( $( "#slider-range-max" ).slider( "value" ) 
  });
  </script>
</head>
<body>

<p>
  <label for="amount">最小的房间数量：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>
<div id="slider-range-max"></div>

</body>
</html>
```

带有固定最小值的范围

固定范围滑块的最小值，用户只能选择最大值。设置 `range` 选项为 "min"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 带有固定最小值的范围</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range-min" ).slider({
      range: "min",
      value: 37,
      min: 1,
      max: 700,
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.value );
      }
    });
    $( "#amount" ).val( "$" + $( "#slider-range-min" ).slider( "val
  });
  </script>
</head>
<body>

<p>
  <label for="amount">最大价格：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-range-min"></div>

</body>
</html>
```

绑定到 select 的滑块

如何绑定一个滑块到一个已有的 select 元素。选择保持可见以便显示变化。当选择改变时，同时更新滑块。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 绑定到 select 的滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var select = $( "#minbeds" );
    var slider = $( "<div id='slider'></div>" ).insertAfter( select
      min: 1,
      max: 6,
      range: "min",
      value: select[ 0 ].selectedIndex + 1,
      slide: function( event, ui ) {
        select[ 0 ].selectedIndex = ui.value - 1;
      }
    });
    $( "#minbeds" ).change(function() {
      slider.slider( "value", this.selectedIndex + 1 );
    });
  });
  </script>
</head>
<body>

<form id="reservation">
  <label for="minbeds">最小的床位数</label>
  <select name="minbeds" id="minbeds">
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
    <option>5</option>
    <option>6</option>
  </select>
</form>

</body>
</html>
```

滑块滚动条

使用滑块来操作页面上内容的定位。本实例中，它是一个能获取值的滚动条。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 滑块滚动条</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .scroll-pane { overflow: auto; width: 99%; float:left; }
  .scroll-content { width: 2440px; float: left; }
  .scroll-content-item { width: 100px; height: 100px; float: left;
  .scroll-bar-wrap { clear: left; padding: 0 4px 0 2px; margin: 0 -
  .scroll-bar-wrap .ui-slider { background: none; border:0; height:
  .scroll-bar-wrap .ui-handle-helper-parent { position: relative; w
  .scroll-bar-wrap .ui-slider-handle { top:.2em; height: 1.5em; }
  .scroll-bar-wrap .ui-slider-handle .ui-icon { margin: -8px auto 0
  </style>
  <script>
  $(function() {
    //滚动面板部分
    var scrollPane = $( ".scroll-pane" ),
      scrollContent = $( ".scroll-content" );

    //创建滑块
    var scrollbar = $( ".scroll-bar" ).slider({
      slide: function( event, ui ) {
        if ( scrollContent.width() > scrollPane.width() ) {
          scrollContent.css( "margin-left", Math.round(
            ui.value / 100 * ( scrollPane.width() - scrollContent.w
          ) + "px" );
        } else {
          scrollContent.css( "margin-left", 0 );
        }
      }
    });

    //追加要处理的图标
    var handleHelper = scrollbar.find( ".ui-slider-handle" )
    .mousedown(function() {
      scrollbar.width( handleHelper.width() );
    })
    .mouseup(function() {
      scrollbar.width( "100%" );
    })
    .append( "<span class='ui-icon ui-icon-grip-dotted-vertical'></
    .wrap( "<div class='ui-handle-helper-parent'></div>" ).parent()

    //由于滑块手柄滚动，改变要隐藏的溢出部分
    scrollPane.css( "overflow", "hidden" );
```

```
    //根据要滚动距离按比例定义滚动条和手柄的尺寸
    function sizeScrollbar() {
      var remainder = scrollContent.width() - scrollPane.width();
      var proportion = remainder / scrollContent.width();
      var handleSize = scrollPane.width() - ( proportion * scrollPa
      scrollbar.find( ".ui-slider-handle" ).css({
        width: handleSize,
        "margin-left": -handleSize / 2
      });
      handleHelper.width( "" ).width( scrollbar.width() - handleSiz
    }

    //基于滚动内容位置，重置滑块的值
    function resetValue() {
      var remainder = scrollPane.width() - scrollContent.width();
      var leftVal = scrollContent.css( "margin-left" ) === "auto" ?
        parseInt( scrollContent.css( "margin-left" ) );
      var percentage = Math.round( leftVal / remainder * 100 );
      scrollbar.slider( "value", percentage );
    }

    //如果滑块是 100%，且窗口变大，则显示内容
    function reflowContent() {
        var showing = scrollContent.width() + parseInt( scrollConte
        var gap = scrollPane.width() - showing;
        if ( gap > 0 ) {
          scrollContent.css( "margin-left", parseInt( scrollContent
        }
    }

    //当缩放窗口时改变手柄的位置
    $( window ).resize(function() {
      resetValue();
      sizeScrollbar();
      reflowContent();
    });
    //初始化滚动条大小
    setTimeout( sizeScrollbar, 10 );//safari 超时
  });
  </script>
</head>
<body>

<div class="scroll-pane ui-widget ui-widget-header ui-corner-all">
  <div class="scroll-content">
    <div class="scroll-content-item ui-widget-header">1</div>
    <div class="scroll-content-item ui-widget-header">2</div>
    <div class="scroll-content-item ui-widget-header">3</div>
    <div class="scroll-content-item ui-widget-header">4</div>
    <div class="scroll-content-item ui-widget-header">5</div>
    <div class="scroll-content-item ui-widget-header">6</div>
    <div class="scroll-content-item ui-widget-header">7</div>
    <div class="scroll-content-item ui-widget-header">8</div>
```

```
      <div class="scroll-content-item ui-widget-header">9</div>
      <div class="scroll-content-item ui-widget-header">10</div>
      <div class="scroll-content-item ui-widget-header">11</div>
      <div class="scroll-content-item ui-widget-header">12</div>
      <div class="scroll-content-item ui-widget-header">13</div>
      <div class="scroll-content-item ui-widget-header">14</div>
      <div class="scroll-content-item ui-widget-header">15</div>
      <div class="scroll-content-item ui-widget-header">16</div>
      <div class="scroll-content-item ui-widget-header">17</div>
      <div class="scroll-content-item ui-widget-header">18</div>
      <div class="scroll-content-item ui-widget-header">19</div>
      <div class="scroll-content-item ui-widget-header">20</div>
    </div>
    <div class="scroll-bar-wrap ui-widget-content ui-corner-bottom">
      <div class="scroll-bar"></div>
    </div>
  </div>

  </body>
  </html>
```

对齐增量

通过把 `step` 选项设置为一个整数来设置滑块值的增量，通常是滑块最大值的除数。默认增量是 1。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 对齐增量</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider" ).slider({
      value:100,
      min: 0,
      max: 500,
      step: 50,
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.value );
      }
    });
    $( "#amount" ).val( "$" + $( "#slider" ).slider( "value" ) );
  });
  </script>
</head>
<body>

<p>
  <label for="amount">捐款金额（$50 增量）：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider"></div>

</body>
</html>
```

垂直的范围滑块

改变范围滑块的方向为垂直的。通过 `.height()` 分配一个高度值，或通过 CSS 设置高度，同时设置 `orientation` 选项为 "vertical"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 垂直的范围滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-range" ).slider({
      orientation: "vertical",
      range: true,
      values: [ 17, 67 ],
      slide: function( event, ui ) {
        $( "#amount" ).val( "$" + ui.values[ 0 ] + " - $" + ui.valu
      }
    });
    $( "#amount" ).val( "$" + $( "#slider-range" ).slider( "values"
      " - $" + $( "#slider-range" ).slider( "values", 1 ) );
  });
  </script>
</head>
<body>

<p>
  <label for="amount">销售目标（百万）：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-range" style="height:250px;"></div>

</body>
</html>
```

垂直的滑块

改变滑块的方向为垂直的。通过 `.height()` 分配一个高度值，或通过 CSS 设置高度，同时设置 `orientation` 选项为 "vertical"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 滑块（Slider） - 垂直的滑块</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#slider-vertical" ).slider({
      orientation: "vertical",
      range: "min",
      min: 0,
      max: 100,
      value: 60,
      slide: function( event, ui ) {
        $( "#amount" ).val( ui.value );
      }
    });
    $( "#amount" ).val( $( "#slider-vertical" ).slider( "value" ) )
  });
  </script>
</head>
<body>

<p>
  <label for="amount">体积：</label>
  <input type="text" id="amount" style="border:0; color:#f6931f; fo
</p>

<div id="slider-vertical" style="height:200px;"></div>

</body>
</html>
```

jQuery UI 实例 - 旋转器（Spinner）

通过向上/向下按钮和箭头键处理，为输入数值增强文本输入功能。

如需了解更多有关 spinner 部件的细节，请查看 API 文档 旋转器部件（Spinner Widget）。

默认功能

默认的旋转器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var spinner = $( "#spinner" ).spinner();

    $( "#disable" ).click(function() {
      if ( spinner.spinner( "option", "disabled" ) ) {
        spinner.spinner( "enable" );
      } else {
        spinner.spinner( "disable" );
      }
    });
    $( "#destroy" ).click(function() {
      if ( spinner.data( "ui-spinner" ) ) {
        spinner.spinner( "destroy" );
      } else {
        spinner.spinner();
      }
    });
    $( "#getvalue" ).click(function() {
      alert( spinner.spinner( "value" ) );
    });
    $( "#setvalue" ).click(function() {
      spinner.spinner( "value", 5 );
    });

    $( "button" ).button();
  });
```

```
  </script>
</head>
<body>

<p>
  <label for="spinner">选择一个值：</label>
  <input id="spinner" name="value">
</p>

<p>
  <button id="disable">切换禁用/启用</button>
  <button id="destroy">切换部件</button>
</p>

<p>
  <button id="getvalue">获取值</button>
  <button id="setvalue">设置值为 5</button>
</p>

</body>
</html>
```

货币

本实例是一个捐款表格，带有货币选择和数量旋转器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 货币</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#currency" ).change(function() {
      $( "#spinner" ).spinner( "option", "culture", $( this ).val()
    });

    $( "#spinner" ).spinner({
      min: 5,
      max: 2500,
      step: 25,
      start: 1000,
      numberFormat: "C"
    });
  });
  </script>
</head>
<body>

<p>
  <label for="currency">要捐款的货币</label>
  <select id="currency" name="currency">
    <option value="en-US">US $</option>
    <option value="de-DE">EUR €</option>
    <option value="ja-JP">YEN ¥</option>
  </select>
</p>
<p>
  <label for="spinner">要捐款的数量：</label>
  <input id="spinner" name="spinner" value="5">
</p>

</body>
</html>
```

小数

本实例是一个小数旋转器。增量设置为 0.01。处理文化变化的代码会读取当前的选择器的值，当改变文化时，会基于新的文化重新设置值的样式。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 小数</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#spinner" ).spinner({
      step: 0.01,
      numberFormat: "n"
    });

    $( "#culture" ).change(function() {
      var current = $( "#spinner" ).spinner( "value" );
      Globalize.culture( $(this).val() );
      $( "#spinner" ).spinner( "value", current );
    });
  });
  </script>
</head>
<body>

<p>
  <label for="spinner">小数旋转器：</label>
  <input id="spinner" name="spinner" value="5.06">
</p>
<p>
  <label for="culture">选择一种用于格式化的文化：</label>
  <select id="culture">
    <option value="en-EN" selected="selected">English</option>
    <option value="de-DE">German</option>
    <option value="ja-JP">Japanese</option>
  </select>
</p>

</body>
</html>
```

地图

谷歌地图集成，使用旋转器来改变纬度和经度。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 地图</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="http://maps.google.com/maps/api/js?sensor=false"></s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    function latlong() {
      return new google.maps.LatLng( $("#lat").val(), $("#lng").val
    }
    function position() {
      map.setCenter( latlong() );
    }
    $( "#lat, #lng" ).spinner({
      step: .001,
      change: position,
      stop: position
    });

    var map = new google.maps.Map( $("#map")[0], {
      zoom: 8,
      center: latlong(),
      mapTypeId: google.maps.MapTypeId.ROADMAP
    });
  });
  </script>
  <style>
  #map {
    width:500px;
    height:500px;
  }
  </style>
</head>
<body>

<label for="lat">纬度</label>
<input id="lat" name="lat" value="44.797">
<br>
<label for="lng">经度</label>
<input id="lng" name="lng" value="-93.278">
```

```
<div id="map"></div>

</body>
</html>
```

溢出

溢出旋转器限制范围从 -10 到 10。对于 10 以上的值，会溢出到 -10，反之亦然。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 溢出</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#spinner" ).spinner({
      spin: function( event, ui ) {
        if ( ui.value > 10 ) {
          $( this ).spinner( "value", -10 );
          return false;
        } else if ( ui.value < -10 ) {
          $( this ).spinner( "value", 10 );
          return false;
        }
      }
    });
  });
  </script>
</head>
<body>

<p>
  <label for="spinner">选择一个值：</label>
  <input id="spinner" name="value">
</p>

</body>
</html>
```

时间

一个扩展自旋转器的自定义部件。使用 全球化（Globalization）插件来解析和输出时间戳，带有自定义的 step 和 page 选项。向上/向下光标用于分钟的递增/递减，向上/向下翻页用于小时的递增/递减。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 旋转器（Spinner） - 时间</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://jqueryui.com/resources/demos/external/jquery.
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="http://jqueryui.com/resources/demos/external/globali
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $.widget( "ui.timespinner", $.ui.spinner, {
    options: {
      // 秒
      step: 60 * 1000,
      // 小时
      page: 60
    },

    _parse: function( value ) {
      if ( typeof value === "string" ) {
        // 已经是一个时间戳
        if ( Number( value ) == value ) {
          return Number( value );
        }
        return +Globalize.parseDate( value );
      }
      return value;
    },

    _format: function( value ) {
      return Globalize.format( new Date(value), "t" );
    }
  });

  $(function() {
    $( "#spinner" ).timespinner();

    $( "#culture" ).change(function() {
      var current = $( "#spinner" ).timespinner( "value" );
      Globalize.culture( $(this).val() );
      $( "#spinner" ).timespinner( "value", current );
    });
  });
  </script>
</head>
```

```
<body>

<p>
  <label for="spinner">时间旋转器：</label>
  <input id="spinner" name="spinner" value="08:30 PM">
</p>
<p>
  <label for="culture">选择一种用于格式化的文化：</label>
  <select id="culture">
    <option value="en-EN" selected="selected">English</option>
    <option value="de-DE">German</option>
  </select>
</p>

</body>
</html>
```

jQuery UI 实例 - 标签页（Tabs）

一种多面板的单内容区，每个面板与列表中的标题相关。

如需了解更多有关 tabs 部件的细节，请查看 API 文档 标签页部件（Tabs Widget）。

默认功能

点击标签页，切换被划分为不同逻辑部分的内容。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs();
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

折叠内容

点击选中的标签页来切换内容的关闭/打开状态。为了启用这个功能，需要设置 `collapsible` 选项为 true。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 折叠内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs({
      collapsible: true
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p><strong>再次点击标签页来关闭内容面板。</strong></p>
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p><strong>再次点击标签页来关闭内容面板。</strong></p>
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p><strong>再次点击标签页来关闭内容面板。</strong></p>
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

通过 Ajax 获取内容

在标签页链接中设置 href 的值来为标签页通过 Ajax 获取外部的内容。当 Ajax 请求在等待响应时，标签页的标签变为 "Loading..."，当加载完成后返回常规的标签。

标签 3 和 4 演示了慢加载和损坏的 AJAX 标签，以及如何处理这些情况下的服务器端的错误。请注意，这两个都要求 web 服务器能解释 PHP。它们在文件系统以外无法工作。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 通过 Ajax 获取内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs({
      beforeLoad: function( event, ui ) {
        ui.jqXHR.error(function() {
          ui.panel.html(
            "不能加载该标签页。如果这不是一个演示。" +
            "我们会尽快修复这个问题。" );
        });
      }
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">预加载</a></li>
    <li><a href="ajax/content1.html">标签 1</a></li>
    <li><a href="ajax/content2.html">标签 2</a></li>
    <li><a href="ajax/content3-slow.php">标签 3 （慢的）</a></li>
    <li><a href="ajax/content4-broken.php">标签 4 （损坏的）</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
</div>

</body>
</html>
```

当鼠标悬停时打开

通过 `event` 选项设置当鼠标悬停时切换各部分的打开/关闭状态。event 的默认值是 "click"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 当鼠标悬停时打开</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs({
      event: "mouseover"
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

简单的操作

简单的标签页添加和移除。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
  <title>jQuery UI 标签页（Tabs） - 简单的操作</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #dialog label, #dialog input { display:block; }
  #dialog label { margin-top: 0.5em; }
  #dialog input, #dialog textarea { width: 95%; }
  #tabs { margin-top: 1em; }
  #tabs li .ui-icon-close { float: left; margin: 0.4em 0.2em 0 0; c
  #add_tab { cursor: pointer; }
  </style>
  <script>
  $(function() {
    var tabTitle = $( "#tab_title" ),
      tabContent = $( "#tab_content" ),
      tabTemplate = "<li><a href='#{href}'>#{label}</a> <span class
      tabCounter = 2;

    var tabs = $( "#tabs" ).tabs();

    // 模态对话框的初始化：自定义按钮和一个重置内部表单的 "close" 回调
    var dialog = $( "#dialog" ).dialog({
      autoOpen: false,
      modal: true,
      buttons: {
        Add: function() {
          addTab();
          $( this ).dialog( "close" );
        },
        Cancel: function() {
          $( this ).dialog( "close" );
        }
      },
      close: function() {
        form[ 0 ].reset();
      }
    });

    // addTab 表单：当提交时调用 addTab 函数，并关闭对话框
    var form = dialog.find( "form" ).submit(function( event ) {
      addTab();
      dialog.dialog( "close" );
      event.preventDefault();
    });

    // 实际的 addTab 函数：使用上面表单的输入添加新的标签页
    function addTab() {
      var label = tabTitle.val() || "Tab " + tabCounter,
        id = "tabs-" + tabCounter,
        li = $( tabTemplate.replace( /#\{href\}/g, "#" + id ).repla
        tabContentHtml = tabContent.val() || "Tab " + tabCounter +
```

```
      tabs.find( ".ui-tabs-nav" ).append( li );
      tabs.append( "<div id='" + id + "'><p>" + tabContentHtml + "<
      tabs.tabs( "refresh" );
      tabCounter++;
    }

    // addTab 按钮：值打开对话框
    $( "#add_tab" )
      .button()
      .click(function() {
        dialog.dialog( "open" );
      });

    // 关闭图标：当点击时移除标签页
    tabs.delegate( "span.ui-icon-close", "click", function() {
      var panelId = $( this ).closest( "li" ).remove().attr( "aria-
      $( "#" + panelId ).remove();
      tabs.tabs( "refresh" );
    });

    tabs.bind( "keyup", function( event ) {
      if ( event.altKey && event.keyCode === $.ui.keyCode.BACKSPACE
        var panelId = tabs.find( ".ui-tabs-active" ).remove().attr(
        $( "#" + panelId ).remove();
        tabs.tabs( "refresh" );
      }
    });
  });
  </script>
</head>
<body>

<div id="dialog" title="Tab data">
  <form>
    <fieldset class="ui-helper-reset">
      <label for="tab_title">标题</label>
      <input type="text" name="tab_title" id="tab_title" value="" c
      <label for="tab_content">内容</label>
      <textarea name="tab_content" id="tab_content" class="ui-widge
    </fieldset>
  </form>
</div>

<button id="add_tab">添加标签页</button>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a> <span class="ui-icon u
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
```

```
</div>

</body>
</html>
```

排序（Sortable）

拖拽上面的标签页来对它们进行重新排序。

只需要简单地调用 `.ui-tabs-nav` 元素上的 `.sortable()`，即可让标签页可排序。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 排序（Sortable）</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    var tabs = $( "#tabs" ).tabs();
    tabs.find( ".ui-tabs-nav" ).sortable({
      axis: "x",
      stop: function() {
        tabs.tabs( "refresh" );
      }
    });
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

底部标签页

通过一些额外的 CSS（用于定位的）和 JS（在元素上放置正确的 class），标签页皆可放置在内容的底部。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 底部标签页</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs();

    // 修改 class
    $( ".tabs-bottom .ui-tabs-nav, .tabs-bottom .ui-tabs-nav > *" )
      .removeClass( "ui-corner-all ui-corner-top" )
      .addClass( "ui-corner-bottom" );

    // 移动导航到底部
    $( ".tabs-bottom .ui-tabs-nav" ).appendTo( ".tabs-bottom" );
  });
  </script>
  <style>
  /* 强制一个高度，这样标签页就不会随着内容高度的改变而改变 */
  #tabs .tabs-spacer { float: left; height: 200px; }
  .tabs-bottom .ui-tabs-nav { clear: left; padding: 0 .2em .2em .2e
  .tabs-bottom .ui-tabs-nav li { top: auto; bottom: 0; margin: 0 .2
  .tabs-bottom .ui-tabs-nav li.ui-tabs-active { margin-top: -1px; p
  </style>
</head>
<body>

<div id="tabs" class="tabs-bottom">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div class="tabs-spacer"></div>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus 
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

垂直的标签页

点击标签页，切换被划分为不同逻辑部分的内容。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 标签页（Tabs） - 垂直的标签页</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#tabs" ).tabs().addClass( "ui-tabs-vertical ui-helper-clear
    $( "#tabs li" ).removeClass( "ui-corner-top" ).addClass( "ui-co
  });
  </script>
  <style>
  .ui-tabs-vertical { width: 55em; }
  .ui-tabs-vertical .ui-tabs-nav { padding: .2em .1em .2em .2em; fl
  .ui-tabs-vertical .ui-tabs-nav li { clear: left; width: 100%; bor
  .ui-tabs-vertical .ui-tabs-nav li a { display:block; }
  .ui-tabs-vertical .ui-tabs-nav li.ui-tabs-active { padding-bottom
  .ui-tabs-vertical .ui-tabs-panel { padding: 1em; float: right; wi
  </style>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <h2>内容标题 1</h2>
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a,
  </div>
  <div id="tabs-2">
    <h2>内容标题 2</h2>
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus
  </div>
  <div id="tabs-3">
    <h2>内容标题 3</h2>
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse pot
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at,
  </div>
</div>

</body>
</html>
```

jQuery UI 实例 - 工具提示框（Tooltip）

可自定义的、可主题化的工具提示框，替代原生的工具提示框。

如需了解更多有关 tooltip 部件的细节，请查看 API 文档 工具提示框部件（Tooltip Widget）。

默认功能

悬停在链接上，或者使用 tab 键循环切换聚焦在每个元素上。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( document ).tooltip();
  });
  </script>
  <style>
  label {
    display: inline-block;
    width: 5em;
  }
  </style>
</head>
<body>

<p><a href="#" title="部件的名称">Tooltips</a> 可被绑定到任意的元素上。当
<p>但是由于它不是一个原生的工具提示框，所以它可以被定义样式。通过 <a href="ht
<p>工具提示框也可以用于表单元素，来显示每个区域中的一些额外的信息。</p>
<p><label for="age">您的年龄：</label><input id="age" title="年龄仅用于
<p>悬停在相应的区域上查看工具提示框。</p>

</body>
</html>
```

自定义样式

悬停在链接上，或者使用 tab 键循环切换聚焦在每个元素上。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 自定义样式</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( document ).tooltip({
      position: {
        my: "center bottom-20",
        at: "center top",
        using: function( position, feedback ) {
          $( this ).css( position );
          $( "<div>" )
            .addClass( "arrow" )
            .addClass( feedback.vertical )
            .addClass( feedback.horizontal )
            .appendTo( this );
        }
      }
    });
  });
  </script>
  <style>
  .ui-tooltip, .arrow:after {
    background: black;
    border: 2px solid white;
  }
  .ui-tooltip {
    padding: 10px 20px;
    color: white;
    border-radius: 20px;
    font: bold 14px "Helvetica Neue", Sans-Serif;
    text-transform: uppercase;
    box-shadow: 0 0 7px black;
  }
  .arrow {
    width: 70px;
    height: 16px;
    overflow: hidden;
    position: absolute;
    left: 50%;
    margin-left: -35px;
    bottom: -16px;
  }
  .arrow.top {
```

```
      top: -16px;
      bottom: auto;
    }
    .arrow.left {
      left: 20%;
    }
    .arrow:after {
      content: "";
      position: absolute;
      left: 20px;
      top: -20px;
      width: 25px;
      height: 25px;
      box-shadow: 6px 5px 9px -9px black;
      -webkit-transform: rotate(45deg);
      -moz-transform: rotate(45deg);
      -ms-transform: rotate(45deg);
      -o-transform: rotate(45deg);
      tranform: rotate(45deg);
    }
    .arrow.top:after {
      bottom: -20px;
      top: auto;
    }
    </style>
  </head>
  <body>

  <p><a href="#" title="部件的名称">Tooltips</a> 可被绑定到任意的元素上。当
  <p>但是由于它不是一个原生的工具提示框，所以它可以被定义样式。通过 <a href="ht
  <p>工具提示框也可以用于表单元素，来显示每个区域中的一些额外的信息。</p>
  <p><label for="age">您的年龄：</label><input id="age" title="年龄仅用
  <p>悬停在相应的区域上查看工具提示框。</p>

  </body>
  </html>
```

自定义动画

本实例演示了如何使用 show 和 hide 选项来自定义动画，也可以使用 open 事件来自定义动画。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 自定义动画</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <script>
  $(function() {
    $( "#show-option" ).tooltip({
      show: {
        effect: "slideDown",
        delay: 250
      }
    });
    $( "#hide-option" ).tooltip({
      hide: {
        effect: "explode",
        delay: 250
      }
    });
    $( "#open-event" ).tooltip({
      show: null,
      position: {
        my: "left top",
        at: "left bottom"
      },
      open: function( event, ui ) {
        ui.tooltip.animate({ top: ui.tooltip.position().top + 10 },
      }
    });
  });
  </script>
</head>
<body>

<p>这里有多种方式自定义工具提示框的动画。</p>
<p>您可以使用 <a id="show-option" href="http://jqueryui.com/demos/to
<a id="hide-option" href="http://jqueryui.com/demos/tooltip/#option
<p>您也可以使用 <a id="open-event" href="http://jqueryui.com/demos/to

</body>
</html>
```

自定义内容

演示如何通过自定义 items 和 content 选项，来组合不同的事件委托工具提示框到一个单一的实例中。

您可能需要与地图工具提示框进行交互，这是未来版本中的一个待实现的功能。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 自定义内容</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .photo {
    width: 300px;
    text-align: center;
  }
  .photo .ui-widget-header {
    margin: 1em 0;
  }
  .map {
    width: 350px;
    height: 350px;
  }
  .ui-tooltip {
    max-width: 350px;
  }
  </style>
  <script>
  $(function() {
    $( document ).tooltip({
      items: "img, [data-geo], [title]",
      content: function() {
        var element = $( this );
        if ( element.is( "[data-geo]" ) ) {
          var text = element.text();
          return "<img class='map' alt='" + text +
            "' src='http://maps.google.com/maps/api/staticmap?" +
            "zoom=11&size=350x350&maptype=terrain&sensor=false&cent
            text + "'>";
        }
        if ( element.is( "[title]" ) ) {
          return element.attr( "title" );
        }
        if ( element.is( "img" ) ) {
          return element.attr( "alt" );
        }
      }
    });
  });
```

```
  </script>
</head>
<body>

<div class="ui-widget photo">
  <div class="ui-widget-header ui-corner-all">
    <h2>圣史蒂芬大教堂（St. Stephen's Cathedral）</h2>
    <h3><a href="http://maps.google.com/maps?q=vienna,+austria&amp;
  </div>
  <a href="http://en.wikipedia.org/wiki/File:Wien_Stefansdom_DSC026
    <img src="img/st-stephens.jpg" alt="圣史蒂芬大教堂（St. Stephen&a
  </a>
</div>

<div class="ui-widget photo">
  <div class="ui-widget-header ui-corner-all">
    <h2>塔桥（Tower Bridge）</h2>
    <h3><a href="http://maps.google.com/maps?q=london,+england&amp;
  </div>
  <a href="http://en.wikipedia.org/wiki/File:Tower_bridge_London_Tw
    <img src="img/tower-bridge.jpg" alt="塔桥（Tower Bridge）" class
  </a>
</div>

<p>所有的图片源自 <a href="http://commons.wikimedia.org/wiki/Main_Pag

</body>
</html>
```

表单

使用下面的按钮来显示帮助文本，或者只需要让鼠标悬停在输入框上或者让输入框获得焦点，即可显示相应输入框的帮助文本。

在 CSS 中定义一个固定的宽度，让同时显示所有的帮助文本时看起来更一致。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 表单</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  label {
    display: inline-block; width: 5em;
  }
```

```
  fieldset div {
    margin-bottom: 2em;
  }
  fieldset .help {
    display: inline-block;
  }
  .ui-tooltip {
    width: 210px;
  }
  </style>
  <script>
  $(function() {
    var tooltips = $( "[title]" ).tooltip();
    $( "<button>" )
      .text( "Show help" )
      .button()
      .click(function() {
        tooltips.tooltip( "open" );
      })
      .insertAfter( "form" );
  });
  </script>
</head>
<body>

<form>
  <fieldset>
    <div>
      <label for="firstname">名字</label>
      <input id="firstname" name="firstname" title="请提供您的名字。"
    </div>
    <div>
      <label for="lastname">姓氏</label>
      <input id="lastname" name="lastname" title="请提供您的姓氏。">
    </div>
    <div>
      <label for="address">地址</label>
      <input id="address" name="address" title="您的家庭或工作地址。">
    </div>
  </fieldset>
</form>

</body>
</html>
```

跟踪鼠标

本实例中的工具提示框是相对于鼠标进行定位的，当鼠标在元素上移动时，它会跟随鼠标移动。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 跟踪鼠标</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  label {
    display: inline-block;
    width: 5em;
  }
  </style>
  <script>
  $(function() {
    $( document ).tooltip({
      track: true
    });
  });
  </script>
</head>
<body>

<p><a href="#" title="部件的名称">Tooltips</a> 可被绑定到任意的元素上。当
<p>但是由于它不是一个原生的工具提示框，所以它可以被定义样式。通过 <a href="ht
<p>工具提示框也可以用于表单元素，来显示每个区域中的一些额外的信息。</p>
<p><label for="age">您的年龄：</label><input id="age" title="年龄仅用于
<p>悬停在相应的区域上查看工具提示框。</p>

</body>
</html>
```

视频播放器

一个虚拟的视频播放器，带有喜欢/分享/统计按钮，每个按钮都带有自定义样式的工具提示框。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 工具提示框（Tooltip） - 视频播放器</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
```

```
<style>
.player {
  width: 500px;
  height: 300px;
  border: 2px groove gray;
  background: rgb(200, 200, 200);
  text-align: center;
  line-height: 300px;
}
.ui-tooltip {
  border: 1px solid white;
  background: rgba(20, 20, 20, 1);
  color: white;
}
.set {
  display: inline-block;
}
.notification {
  position: absolute;
  display: inline-block;
  font-size: 2em;
  padding: .5em;
  box-shadow: 2px 2px 5px -2px rgba(0,0,0,0.5);
}
</style>
<script>
$(function() {
  function notify( input ) {
    var msg = "已选择 " + $.trim( input.data( "tooltip-title" ) |
    $( "<div>" )
      .appendTo( document.body )
      .text( msg )
      .addClass( "notification ui-state-default ui-corner-bottom'
      .position({
        my: "center top",
        at: "center top",
        of: window
      })
      .show({
        effect: "blind"
      })
      .delay( 1000 )
      .hide({
        effect: "blind",
        duration: "slow"
      }, function() {
        $( this ).remove();
      });
  }

  $( "button" ).each(function() {
    var button = $( this ).button({
      icons: {
```

```
          primary: $( this ).data( "icon" )
        },
        text: !!$( this ).attr( "title" )
      });
      button.click(function() {
        notify( button );
      });
    });
    $( ".set" ).buttonset({
      items: "button"
    });

    $( document ).tooltip({
      position: {
        my: "center top",
        at: "center bottom+5",
      },
      show: {
        duration: "fast"
      },
      hide: {
        effect: "hide"
      }
    });
  });
  </script>
</head>
<body>

<div class="player">这里是视频 (HTML5?)</div>
<div class="tools">
  <span class="set">
    <button data-icon="ui-icon-circle-arrow-n" title="我喜欢这个视频"
    <button data-icon="ui-icon-circle-arrow-s">我不喜欢这个视频</butt
  </span>
  <div class="set">
    <button data-icon="ui-icon-circle-plus" title="添加到播放列表">添
    <button class="menu" data-icon="ui-icon-triangle-1-s">添加到收藏
  </div>
  <button title="分享这个视频">分享</button>
  <button data-icon="ui-icon-alert">标记为不恰当</button>
</div>

</body>
</html>
```

jQuery UI 实例 - 特效（Effect）

对一个元素应用动画特效。

如需了解更多有关 `.effect()` 方法的细节，请查看 API 文档 .effect()。

.effect() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .effect() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .toggler { width: 500px; height: 200px; position: relative; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px; height: 135px; padding: 0.4em; positior
    #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
    .ui-effects-transfer { border: 2px dotted gray; }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 0 };
      } else if ( selectedEffect === "transfer" ) {
        options = { to: "#button", className: "ui-effects-transfer'
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 200, height: 60 } };
      }

      // 运行特效
      $( "#effect" ).effect( selectedEffect, options, 500, callbacl
    };
```

```
    // 回调函数
    function callback() {
      setTimeout(function() {
        $( "#effect" ).removeAttr( "style" ).hide().fadeIn();
      }, 1000 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">特效（Effect）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fade">淡入淡出特效（Fade Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
  <option value="transfer">转移特效（Transfer Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

Easing 演示

本实例使用 HTML Canvas 元素，绘制了 jQuery UI 提供的所有 easings。 点击每个图可查看该 easing 的行为。。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - Easing 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .graph {
    float: left;
    margin-left: 10px;
  }
  </style>
  <script>
  $(function() {
    if ( !$( "<canvas>" )[0].getContext ) {
      $( "<div>" ).text(
        "您的浏览器不支持 canvas，本演示需要在支持 canvas 的浏览器下进行。"
      ).appendTo( "#graphs" );
      return;
    }

    var i = 0,
      width = 100,
      height = 100;

    $.each( $.easing, function( name, impl ) {
      var graph = $( "<div>" ).addClass( "graph" ).appendTo( "#grap
        text = $( "<div>" ).text( ++i + ". " + name ).appendTo( gra
        wrap = $( "<div>" ).appendTo( graph ).css( 'overflow', 'hid
        canvas = $( "<canvas>" ).appendTo( wrap )[ 0 ];

      canvas.width = width;
      canvas.height = height;
      var drawHeight = height * 0.8,
        cradius = 10;
        ctx = canvas.getContext( "2d" );
      ctx.fillStyle = "black";

      // 绘制背景
      ctx.beginPath();
      ctx.moveTo( cradius, 0 );
      ctx.quadraticCurveTo( 0, 0, 0, cradius );
      ctx.lineTo( 0, height - cradius );
      ctx.quadraticCurveTo( 0, height, cradius, height );
      ctx.lineTo( width - cradius, height );
      ctx.quadraticCurveTo( width, height, width, height - cradius
```

```
        ctx.lineTo( width, 0 );
        ctx.lineTo( cradius, 0 );
        ctx.fill();

        // 绘制底线
        ctx.strokeStyle = "#555";
        ctx.beginPath();
        ctx.moveTo( width * 0.1, drawHeight + .5 );
        ctx.lineTo( width * 0.9, drawHeight + .5 );
        ctx.stroke();

        // 绘制顶线
        ctx.strokeStyle = "#555";
        ctx.beginPath();
        ctx.moveTo( width * 0.1, drawHeight * .3 - .5 );
        ctx.lineTo( width * 0.9, drawHeight * .3 - .5 );
        ctx.stroke();

        // 绘制 easing
        ctx.strokeStyle = "white";
        ctx.beginPath();
        ctx.lineWidth = 2;
        ctx.moveTo( width * 0.1, drawHeight );
        $.each( new Array( width ), function( position ) {
          var state = position / width,
            val = impl( state, position, 0, 1, width );
          ctx.lineTo( position * 0.8 + width * 0.1,
            drawHeight - drawHeight * val * 0.7 );
        });
        ctx.stroke();

        // 点击时动态改变
        graph.click(function() {
          wrap
            .animate( { height: "hide" }, 2000, name )
            .delay( 800 )
            .animate( { height: "show" }, 2000, name );
        });

        graph.width( width ).height( height + text.height() + 10 );
      });
    });
  </script>
</head>
<body>

<div id="graphs"></div>

</body>
</html>
```

jQuery UI 实例 - 显示（Show）

使用自定义效果来显示匹配的元素。

如需了解更多有关 `.show()` 方法的细节，请查看 API 文档 .show()。

.show() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .show() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { width: 240px; height: 135px; padding: 0.4em; position:
  #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 100 };
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 280, height: 185 } };
      }

      // 运行特效
      $( "#effect" ).show( selectedEffect, options, 500, callback )
    };

    // 回调函数
    function callback() {
      setTimeout(function() {
```

```
        $( "#effect:visible" ).removeAttr( "style" ).fadeOut();
      }, 1000 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });

    $( "#effect" ).hide();
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">显示（Show）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

 <select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 隐藏（Hide）

使用自定义效果来隐藏匹配的元素。

如需了解更多有关 `.hide()` 方法的细节，请查看 API 文档 .hide()。

.hide() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .hide() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { width: 240px; height: 135px; padding: 0.4em; position:
  #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 0 };
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 200, height: 60 } };
      }

      // 运行特效
      $( "#effect" ).hide( selectedEffect, options, 1000, callback
    };

    // 回调函数
    function callback() {
      setTimeout(function() {
```

```
        $( "#effect" ).removeAttr( "style" ).hide().fadeIn();
      }, 1000 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">隐藏（Hide）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 切换（Toggle）

使用自定义效果来显示或隐藏匹配的元素。

如需了解更多有关 `.toggle()` 方法的细节，请查看 API 文档 .toggle()。

.toggle() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .toggle() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler {
    width: 500px;
    height: 200px;
  }
  #button {
    padding: .5em 1em;
    text-decoration: none;
  }
  #effect {
    position: relative;
    width: 240px;
    height: 135px;
    padding: 0.4em;
  }
  #effect h3 {
    margin: 0;
    padding: 0.4em;
    text-align: center;
  }
  </style>
  <script>
  $(function() {
    // 运行当前选中的特效
    function runEffect() {
      // 从中获取特效类型
      var selectedEffect = $( "#effectTypes" ).val();

      // 大多数的特效类型默认不需要传递选项
```

```
      var options = {};
      // 一些特效带有必需的参数
      if ( selectedEffect === "scale" ) {
        options = { percent: 0 };
      } else if ( selectedEffect === "size" ) {
        options = { to: { width: 200, height: 60 } };
      }

      // 运行特效
      $( "#effect" ).toggle( selectedEffect, options, 500 );
    };

    // 根据选择菜单值设置特效
    $( "#button" ).click(function() {
      runEffect();
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">切换（Toggle）</h3>
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<select name="effects" id="effectTypes">
  <option value="blind">百叶窗特效（Blind Effect）</option>
  <option value="bounce">反弹特效（Bounce Effect）</option>
  <option value="clip">剪辑特效（Clip Effect）</option>
  <option value="drop">降落特效（Drop Effect）</option>
  <option value="explode">爆炸特效（Explode Effect）</option>
  <option value="fold">折叠特效（Fold Effect）</option>
  <option value="highlight">突出特效（Highlight Effect）</option>
  <option value="puff">膨胀特效（Puff Effect）</option>
  <option value="pulsate">跳动特效（Pulsate Effect）</option>
  <option value="scale">缩放特效（Scale Effect）</option>
  <option value="shake">震动特效（Shake Effect）</option>
  <option value="size">尺寸特效（Size Effect）</option>
  <option value="slide">滑动特效（Slide Effect）</option>
</select>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 添加 Class（Add Class）

当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

如需了解更多有关 `.addClass()` 方法的细节，请查看 API 文档 .addClass()。

.addClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .addClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .toggler { width: 500px; height: 200px; position: relative; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px;  padding: 1em;  font-size: 1.2em; borde
    .newClass { text-indent: 40px; letter-spacing: .4em; width: 41(
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function() {
      $( "#effect" ).addClass( "newClass", 1000, callback );
      return false;
    });

    function callback() {
      setTimeout(function() {
        $( "#effect" ).removeClass( "newClass" );
      }, 1500 );
    }
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-corner-all">
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 移除 Class（Remove Class）

当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

如需了解更多有关 `.removeClass()` 方法的细节，请查看 API 文档 .removeClass()。

.removeClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .removeClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; position: relative; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { position: relative;  width: 240px;  padding: 1em;  lett
  .newClass { text-indent: 40px; letter-spacing: .4em; width: 410px
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function() {
      $( "#effect" ).removeClass( "newClass", 1000, callback );
      return false;
    });

    function callback() {
      setTimeout(function() {
        $( "#effect" ).addClass( "newClass" );
      }, 1500 );
    }
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="newClass ui-corner-all">
    Etiam libero neque, luctus a, eleifend nec, semper at, lorem. S
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 切换 Class（Toggle Class）

当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

如需了解更多有关 `.toggleClass()` 方法的细节，请查看 API 文档 .toggleClass()。

.toggleClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .toggleClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; position: relative; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect {position: relative;  width: 240px;  padding: 1em; letter
  .newClass { text-indent: 40px; letter-spacing: .4em; width: 410px
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function() {
      $( "#effect" ).toggleClass( "newClass", 1000 );
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="newClass ui-corner-all">
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 转换 Class（Switch Class）

当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。

如需了解更多有关 `.switchClass()` 方法的细节，请查看 API 文档 .switchClass()。

.switchClass() 演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - .switchClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .toggler { width: 500px; height: 200px; position: relative; }
  #button { padding: .5em 1em; text-decoration: none; }
  #effect { position: relative; }
  .newClass { width: 240px;  padding: 1em; letter-spacing: 0; font-
  .anotherNewClass { text-indent: 40px; letter-spacing: .4em; width
  </style>
  <script>
  $(function() {
    $( "#button" ).click(function(){
      $( ".newClass" ).switchClass( "newClass", "anotherNewClass", 
      $( ".anotherNewClass" ).switchClass( "anotherNewClass", "newC
      return false;
    });
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="newClass ui-corner-all">
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
  </div>
</div>
<a href="#" id="button" class="ui-state-default ui-corner-all">运行

</body>
</html>
```

jQuery UI 实例 - 颜色动画（Color Animation）

使用 .animate() 实现颜色动画效果。

如需了解更多有关颜色动画（Color Animation）的细节，请查看 API 文档 颜色动画（Color Animation）。

jQuery UI 捆绑了 jQuery Color 插件，jQuery Color 插件提供了颜色动画及其他许多与颜色相关的实用功能。

动画（Animation）演示

点击按钮预览特效。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 特效 - 动画（Animation）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
    .toggler { width: 500px; height: 200px; position: relative; }
    #button { padding: .5em 1em; text-decoration: none; }
    #effect { width: 240px; height: 135px; padding: 0.4em; position
    #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
  </style>
  <script>
  $(function() {
    var state = true;
    $( "#button" ).click(function() {
      if ( state ) {
        $( "#effect" ).animate({
          backgroundColor: "#aa0000",
          color: "#fff",
          width: 500
        }, 1000 );
      } else {
        $( "#effect" ).animate({
          backgroundColor: "#fff",
          color: "#000",
          width: 240
        }, 1000 );
      }
      state = !state;
    });
```

```
  });
  </script>
</head>
<body>

<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">动画（Animation）</h3
    <p>
      Etiam libero neque, luctus a, eleifend nec, semper at, lorem.
    </p>
  </div>
</div>

<a href="#" id="button" class="ui-state-default ui-corner-all">切换

</body>
</html>
```

jQuery UI 实例 - 定位（Position）

相对窗口、文档、锚、光标/鼠标等元素定位一个元素。

如需了解更多有关 `.position()` 方法的细节，请查看 API 文档 .position()。

默认功能

使用表单控件配置位置，或者拖拽被定位的元素来修改它的偏移量。向四周拖拽父元素来查看碰撞检测。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 定位（Position） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  #parent {
    width: 60%;
    height: 40px;
    margin: 10px auto;
    padding: 5px;
    border: 1px solid #777;
    background-color: #fbca93;
    text-align: center;
  }
  .positionable {
    position: absolute;
    display: block;
    right: 0;
    bottom: 0;
    background-color: #bcd5e6;
    text-align: center;
  }
  #positionable1 {
    width: 75px;
    height: 75px;
  }
  #positionable2 {
    width: 120px;
    height: 40px;
  }
  select, input {
    margin-left: 15px;
```

```
  }
  </style>
  <script>
  $(function() {
    function position() {
      $( ".positionable" ).position({
        of: $( "#parent" ),
        my: $( "#my_horizontal" ).val() + " " + $( "#my_vertical" )
        at: $( "#at_horizontal" ).val() + " " + $( "#at_vertical" )
        collision: $( "#collision_horizontal" ).val() + " " + $( "#
      });
    }

    $( ".positionable" ).css( "opacity", 0.5 );

    $( "select, input" ).bind( "click keyup change", position );

    $( "#parent" ).draggable({
      drag: position
    });

    position();
  });
  </script>
</head>
<body>

<div id="parent">
  <p>
  这是父元素的位置。
  </p>
</div>

<div class="positionable" id="positionable1">
  <p>
  to position
  </p>
</div>

<div class="positionable" id="positionable2">
  <p>
  to position 2
  </p>
</div>

<div style="padding: 20px; margin-top: 75px;">
  定位...
  <div style="padding-bottom: 20px;">
    <b>my:</b>
    <select id="my_horizontal">
      <option value="left">left</option>
      <option value="center">center</option>
      <option value="right">right</option>
```

```
      </select>
      <select id="my_vertical">
        <option value="top">top</option>
        <option value="center">center</option>
        <option value="bottom">bottom</option>
      </select>
    </div>
    <div style="padding-bottom: 20px;">
      <b>at:</b>
      <select id="at_horizontal">
        <option value="left">left</option>
        <option value="center">center</option>
        <option value="right">right</option>
      </select>
      <select id="at_vertical">
        <option value="top">top</option>
        <option value="center">center</option>
        <option value="bottom">bottom</option>
      </select>
    </div>
    <div style="padding-bottom: 20px;">
      <b>collision:</b>
      <select id="collision_horizontal">
        <option value="flip">flip</option>
        <option value="fit">fit</option>
        <option value="flipfit">flipfit</option>
        <option value="none">none</option>
      </select>
      <select id="collision_vertical">
        <option value="flip">flip</option>
        <option value="fit">fit</option>
        <option value="flipfit">flipfit</option>
        <option value="none">none</option>
      </select>
    </div>
  </div>

</body>
</html>
```

图像循环

一个照片浏览器的原型，使用 Position 分别把图片定为在左边、中间、右边，然后循环显示。使用顶部的链接来循环图像，或者在图像的左侧或右侧点击来循环图像。请注意，当调整窗口大小时，会重新定位图像。

```
<!doctype html>
<html lang="en">
<head>
```

```
  <meta charset="utf-8">
  <title>jQuery UI 定位（Position） - 图像循环</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  body {
    margin: 0;
  }
  #container {
    overflow: hidden;
    position: relative;
    height: 400px;
  }

  img {
    position: absolute;
  }
  </style>
  <script>
  $(function() {
    // 重构部件，去除这些插件方法
    $.fn.left = function( using ) {
      return this.position({
        my: "right middle",
        at: "left+25 middle",
        of: "#container",
        collision: "none",
        using: using
      });
    };
    $.fn.right = function( using ) {
      return this.position({
        my: "left middle",
        at: "right-25 middle",
        of: "#container",
        collision: "none",
        using: using
      });
    };
    $.fn.center = function( using ) {
      return this.position({
        my: "center middle",
        at: "center middle",
        of: "#container",
        using: using
      });
    };

    $( "img:eq(0)" ).left();
    $( "img:eq(1)" ).center();
    $( "img:eq(2)" ).right();
```

```
    function animate( to ) {
      $( this ).stop( true, false ).animate( to );
    }
    function next( event ) {
      event.preventDefault();
      $( "img:eq(2)" ).center( animate );
      $( "img:eq(1)" ).left( animate );
      $( "img:eq(0)" ).right().appendTo( "#container" );
    }
    function previous( event ) {
      event.preventDefault();
      $( "img:eq(0)" ).center( animate );
      $( "img:eq(1)" ).right( animate );
      $( "img:eq(2)" ).left().prependTo( "#container" );
    }
    $( "#previous" ).click( previous );
    $( "#next" ).click( next );

    $( "img" ).click(function( event ) {
      $( "img" ).index( this ) === 0 ? previous( event ) : next( ev
    });

    $( window ).resize(function() {
      $( "img:eq(0)" ).left( animate );
      $( "img:eq(1)" ).center( animate );
      $( "img:eq(2)" ).right( animate );
    });
  });
  </script>
</head>
<body>

<div id="container">
  <img src="img/earth.jpg" width="458" height="308" alt="earth">
  <img src="img/flight.jpg" width="512" height="307" alt="flight">
  <img src="img/rocket.jpg" width="300" height="353" alt="rocket">

  <a id="previous" href="#">上一个</a>
  <a id="next" href="#">下一个</a>
</div>

</body>
</html>
```

jQuery UI 实例 - 部件库（Widget Factory）

使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

如需了解更多有关部件库（Widget Factory）的细节，请查看 API 文档 部件库（Widget Factory）。

默认功能

该演示展示了一个简单的使用部件库（jquery.ui.widget.js）创建的自定义的小部件。

三个区块是以不同的方式初始化的。点击改变他们的背景颜色。查看源码及注释理解工作原理。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI 部件库（Widget Factory） - 默认功能</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="http://jqueryui.com/resources/demos/
  <style>
  .custom-colorize {
    font-size: 20px;
    position: relative;
    width: 75px;
    height: 75px;
  }
  .custom-colorize-changer {
    font-size: 10px;
    position: absolute;
    right: 0;
    bottom: 0;
  }
  </style>
  <script>
  $(function() {
    // 部件定义，其中 "custom" 是命名空间，"colorize" 是部件名称
    $.widget( "custom.colorize", {
      // 默认选项
      options: {
        red: 255,
        green: 0,
        blue: 0,
```

```
      // 回调
      change: null,
      random: null
    },

    // 构造函数
    _create: function() {
      this.element
        // 添加一个主题化的 class
        .addClass( "custom-colorize" )
        // 防止双击来选择文本
        .disableSelection();

      this.changer = $( "<button>", {
        text: "改变",
        "class": "custom-colorize-changer"
      })
      .appendTo( this.element )
      .button();

      // 绑定 changer 按钮上的 click 事件到 random 方法
      this._on( this.changer, {
        // 当小部件被禁用时，_on 不会调用 random
        click: "random"
      });
      this._refresh();
    },

    // 当创建及之后改变选项时调用
    _refresh: function() {
      this.element.css( "background-color", "rgb(" +
        this.options.red +"," +
        this.options.green + "," +
        this.options.blue + ")"
      );

      // 触发一个回调/事件
      this._trigger( "change" );
    },

    // 一个改变颜色为随机值的公共方法
    // 可通过 .colorize( "random" ) 直接调用
    random: function( event ) {
      var colors = {
        red: Math.floor( Math.random() * 256 ),
        green: Math.floor( Math.random() * 256 ),
        blue: Math.floor( Math.random() * 256 )
      };

      // 触发一个事件，检查是否已取消
      if ( this._trigger( "random", event, colors ) !== false ) 
        this.option( colors );
      }
```

```
      },

      // 自动移除通过 _on 绑定的事件
      // 在这里重置其他的修改
      _destroy: function() {
        // 移除生成的元素
        this.changer.remove();

        this.element
          .removeClass( "custom-colorize" )
          .enableSelection()
          .css( "background-color", "transparent" );
      },

      // _setOptions 是通过一个带有所有改变的选项的哈希来调用的
      // 当改变选项时总是刷新
      _setOptions: function() {
        // _super 和 _superApply
        this._superApply( arguments );
        this._refresh();
      },

      // _setOption 是为每个独立的改变的选项调用的
      _setOption: function( key, value ) {
        // 防止无效的颜色值
        if ( /red|green|blue/.test(key) && (value < 0 || value > 25
          return;
        }
        this._super( key, value );
      }
    });

    // 通过默认选项进行初始化
    $( "#my-widget1" ).colorize();

    // 通过两个自定义的选项进行初始化
    $( "#my-widget2" ).colorize({
      red: 60,
      blue: 60
    });

    // 通过自定义的 green 值和一个只允许颜色足够绿的随机的回调进行初始化
    $( "#my-widget3" ).colorize( {
      green: 128,
      random: function( event, ui ) {
        return ui.green > 128;
      }
    });

    // 点击切换 enabled/disabled
    $( "#disable" ).click(function() {
      // 为每个小部件使用自定义的选择器来找到所有的实例
      // 所有的实例一起切换，所以我们可以从第一个开始检查状态
```

```
      if ( $( ":custom-colorize" ).colorize( "option", "disabled" ) 
        $( ":custom-colorize" ).colorize( "enable" );
      } else {
        $( ":custom-colorize" ).colorize( "disable" );
      }
    });

    // 在初始化后，点击设置选项
    $( "#black" ).click( function() {
      $( ":custom-colorize" ).colorize( "option", {
        red: 0,
        green: 0,
        blue: 0
      });
    });
  });
  </script>
</head>
<body>

<div>
  <div id="my-widget1">改变颜色</div>
  <div id="my-widget2">改变颜色</div>
  <div id="my-widget3">改变颜色</div>
  <button id="disable">切换 disabled 选项</button>
  <button id="black">改为黑色</button>
</div>

</body>
</html>
```

jQuery UI API 参考

jQuery UI API 类别 - 特效（Effects）

jQuery UI 在jQuery 内置的特效上添加了一些功能。jQuery UI 支持颜色动画和 Class 转换，同时也提供了一些额外的 Easings。另外，提供了一套完整的定制特效，供显示和隐藏元素时或者只是添加一些视觉显示时使用。

API	描述	也属于类别
.addClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。	特效核心（Effects Core） | 方法重载（Method Overrides）
百叶窗特效（Blind Effect）	百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。	
反弹特效（Bounce Effect）	反弹特效（Bounce Effect）反弹一个元素。当与隐藏或显示一起使用时，最后一次或第一次反弹会呈现淡入/淡出效果。	
剪辑特效（Clip Effect）	剪辑特效（Clip Effect）通过垂直或水平方向夹剪元素来隐藏或显示一个元素。	
颜色动画（Color Animation）	使用 .animate() 实现颜色动画效果。	
降落特效（Drop Effect）	降落特效（Drop Effect）通过单个方向滑动的淡入淡出来隐藏或显示一个元素。	
Easings	Easing 函数指定动画在不同点上的行进速度。	
.effect()	对一个元素应用动画特效。	特效核心（Effects Core） | 方法（Method）
爆炸特效（Explode Effect）	爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。	
淡入淡出特效（Fade Effect）	淡入淡出特效（Fade Effect）通过淡入淡出元素来隐藏或显示一个元素。	

（Fold Effect）	素来隐藏或显示一个元素。	
.hide()	使用自定义效果来隐藏匹配的元素。	特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Method）
突出特效（Highlight Effect）	突出特效（Highlight Effect）通过首先改变背景颜色来隐藏或显示一个元素。	
膨胀特效（Puff Effect）	通过在缩放元素的同时隐藏元素来创建膨胀特效（Puff Effect）。	
跳动特效（Pulsate Effect）	跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。	
.removeClass()	当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。	特效核心（Effects Core） | 方法重载（Method Overrides）
缩放特效（Scale Effect）	按照某个百分比缩放元素。	
震动特效（Shake Effect）	垂直或水平方向多次震动元素。	
.show()	使用自定义效果来显示匹配的元素。	特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Method）
尺寸特效（Size Effect）	调整元素尺寸到指定宽度和高度。	
滑动特效（Slide Effect）	把元素滑动出视区。	
.switchClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。	特效核心（Effects Core）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效核心（Effects Core） | 方法重载（Method

	的元素。	Overrides） | 方法（Method）
.toggleClass()	当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。	特效核心（Effects Core） | 方法重载（Method Overrides）
转移特效（Transfer Effect）	把一个元素的轮廓转移到另一个元素。	

jQuery UI API - .addClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。

返回：jQuery

```
.addClass( className [, duration ] [, easing ] [, complete ] )
```

参数	类型	描述	默认值
className	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.addClass( className [, options ] )
```

参数	类型	描述
className	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值：`swing`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。

- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.addClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画的结尾改变。

该插件扩展自 jQuery 内置的 .addClass() 方法。如果 jQuery UI 未加载，调用 `.addClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

添加 class "big-blue" 到匹配的元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.addClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div></div>

<script>
$( "div" ).click(function() {
  $( this ).addClass( "big-blue", 1000, "easeOutBounce" );
});
</script>

</body>
</html>
```

jQuery UI API - 百叶窗特效（Blind Effect）

所属类别

特效（Effects）

用法

描述：百叶窗特效（Blind Effect）通过将元素包裹在一个容器内，采用"拉百叶窗"效果来隐藏或显示元素。

```
blind
```

参数	类型	描述	默认值
direction	String	隐藏元素时拉动百叶窗的方向，显示元素时拉动百叶窗的方向。	

可能的值：up、down、left、right、vertical、horizontal。 | "up" |

容器应用 `overflow: hidden` 时，高度的变化会影响到元素的可见性。

实例

使用百叶窗特效（Blind Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>百叶窗特效（Blind Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "blind" );
});
</script>

</body>
</html>
```

jQuery UI API - 反弹特效（Bounce Effect）

所属类别

特效（Effects）

用法

描述：反弹特效（Bounce Effect）反弹一个元素。当与隐藏或显示一起使用时，最后一次或第一次反弹会呈现淡入/淡出效果。

```
bounce
```

参数	类型	描述	默认值
distance	Number	最大的反弹距离，以像素为单位。	20
times	Integer	元素反弹的次数。当隐藏或显示时，会为淡入/淡出效果添加半个反弹。	5

实例

使用反弹特效（Bounce Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>反弹特效（Bounce Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "bounce", { times: 3 }, "slow" );
});
</script>

</body>
</html>
```

jQuery UI API - 剪辑特效（Clip Effect）

所属类别

特效（Effects）

用法

描述：剪辑特效（Clip Effect）通过垂直或水平方向夹剪元素来隐藏或显示一个元素。

```
clip
```

参数	类型	描述	默认值
direction	String	剪辑特效隐藏或显示元素的方向。	

vertical 剪辑上下边缘，horizontal 剪辑左右边缘。 | "vertical" |

实例

使用剪辑特效（Clip Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>剪辑特效（Clip Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "clip" );
});
</script>

</body>
</html>
```

jQuery UI API - 颜色动画（Color Animation）

jQuery UI 特效核心添加了使用 `rgb()` 、 `rgba()` 、十六进制值或者颜色名比如 `"aqua"` 来动态改变 color 属性的功能。只需要包含 jQuery UI 特效核心文件， `.animate()` 就会支持颜色。

支持下列属性：

- `backgroundColor`
- `borderBottomColor`
- `borderLeftColor`
- `borderRightColor`
- `borderTopColor`
- `color`
- `columnRuleColor`
- `outlineColor`
- `textDecorationColor`
- `textEmphasisColor`

对颜色动画的支持来自 jQuery Color 插件。Color 插件提供了一些用于颜色的函数。如需查看完整文档，请访问 jQuery Color 文档。

Class 动画（Class Animations）

虽然可以直接对 color 属性进行动画化，但是通常采用另一种更好的方法，即在一个 class 中包含样式。jQuery UI 提供了一些动态添加或去除 CSS 类的方法，分别是 `.addClass()` 、 `.removeClass()` 、 `.toggleClass()` 和 `.switchClass()` 。这些方法将自动确定哪些属性需要改变，哪些需要应用适当的动画。

实例

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>颜色动画（Color Animation）演示</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/the
  <style>
  #elem {
    color: #006;
    background-color: #aaa;
    font-size: 25px;
    padding: 1em;
    text-align: center;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></scr
</head>
<body>

<div id="elem">颜色动画</div>
<button id="toggle">改变颜色</button>

<script>
$( "#toggle" ).click(function() {
  $( "#elem" ).animate({
    color: "green",
    backgroundColor: "rgb( 20, 20, 20 )"
  });
});
</script>

</body>
</html>
```

jQuery UI API - 降落特效（Drop Effect）

所属类别

特效（Effects）

用法

描述：降落特效（Drop Effect）通过单个方向滑动的淡入淡出来隐藏或显示一个元素。

```
drop
```

参数	类型	描述	默认值
direction	String	隐藏元素时元素降落的方向，显示元素时元素出现的方向。	

可能的值：up、down、left、right。 | "left" |

实例

使用降落特效（Blind Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>降落特效（Drop Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "drop" );
});
</script>

</body>
</html>
```

jQuery UI API - Easings

Easing 函数指定动画在不同点上的行进速度。jQuery 核心带有两个 easings：一个是 `linear`，整个动画以一个不变的速度行进；另一个是 `swing`（jQuery 核心默认的 easing），行进速度在动画开始和结束时比在动画中间时稍慢。jQuery UI 提供了一些额外的 easing 函数，范围从摆动行为上的变化到定制特效，比如弹跳。

一些 easings 会在动画中产生负值。根据动画的不同属性，实际值可能为零。例如，您可以把 `left` 取为负值，但是不能把 `height` 或 `opacity` 取为负值。

想要更好地理解一个 easing 如何影响一个动画，需多花时间研究方程图。请看下面所列出的 jQuery UI 中所有可用动画的曲线图。

jQuery UI API - .effect()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法（Methods）

用法

描述：对一个元素应用动画特效。

返回：jQuery

```
.effect( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.effect( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

`.effect()` 方法对一个元素应用了一个命名的动画 特效 。许多特效也支持显示或隐藏模式，这些可通过 `.show()` 、 `.hide()` 和 `.toggle()` 方法来完成。

实例

对一个 div 应用反弹特效（Bounce Effect）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.effect() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方应用特效。</p>
<div></div>

<script>
$( document ).click(function() {
  $( "div" ).effect( "bounce", "slow" );
});
</script>

</body>
</html>
```

jQuery UI API - 爆炸特效（Explode Effect）

所属类别

特效（Effects）

用法

描述：爆炸特效（Explode Effect）通过把元素裂成碎片来隐藏或显示一个元素。

```
explode
```

参数	类型	描述	默认值
pieces	Integer	爆炸裂开的碎片数目，应该是个平方数，任何其他值被舍入到最近的平方数。	9

实例

使用爆炸特效（Explode Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>爆炸特效（Explode Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "explode" );
});
</script>

</body>
</html>
```

jQuery UI API - 淡入淡出特效（Fade Effect）

所属类别

特效（Effects）

用法

描述：淡入淡出特效（Fade Effect）通过淡入淡出元素来隐藏或显示一个元素。

```
fade
```

实例

使用淡入淡出特效（Fade Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>淡入淡出特效（Fade Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "fade" );
});
</script>

</body>
</html>
```

jQuery UI API - 折叠特效（Fold Effect）

所属类别

特效（Effects）

用法

描述：折叠特效（Fold Effect）通过折叠元素来隐藏或显示一个元素。

```
fold
```

参数	类型	描述	默认值
size	Number 或 String	被折叠元素的尺寸。	15
horizFirst	Boolean	当折叠时是否先进行水平方向的折叠。请记得，显示的时候与隐藏的时候顺序相反。	false

实例

使用折叠特效（Fold Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>折叠特效（Fold Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "fold" );
});
</script>

</body>
</html>
```

jQuery UI API - .hide()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Methods）

用法

描述：使用自定义效果来隐藏匹配的元素。

返回：jQuery

```
.hide( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.hide( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串

表示的队列中。

该插件扩展自 jQuery 内置的 .hide() 方法。如果 jQuery UI 未加载，调用 `.hide()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

使用降落特效（Drop Effect）隐藏一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.hide() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>隐藏 div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).hide( "drop", { direction: "down" }, "slow" );
});
</script>

</body>
</html>
```

jQuery UI API - 突出特效（Highlight Effect）

所属类别

[特效（Effects）]

用法

描述：突出特效（Highlight Effect）通过首先改变背景颜色来隐藏或显示一个元素。

```
highlight
```

参数	类型	描述	默认值
color	String	动画期间使用的背景颜色。	"#ffff99"

实例

使用突出特效（Highlight Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>突出特效（Highlight Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "highlight" );
});
</script>

</body>
</html>
```

jQuery UI API - 膨胀特效（Puff Effect）

所属类别

特效（Effects）

用法

描述：通过在缩放元素的同时隐藏元素来创建膨胀特效（Puff Effect）。

```
puff
```

参数	类型	描述	默认值
percent	Number	要缩放到的百分比。	150

实例

使用膨胀特效（Puff Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>膨胀特效（Puff Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "puff" );
});
</script>

</body>
</html>
```

jQuery UI API - 跳动特效（Pulsate Effect）

所属类别

特效（Effects）

用法

描述：跳动特效（Pulsate Effect）通过跳动来隐藏或显示一个元素。

```
pulsate
```

参数	类型	描述	默认值
times	Integer	元素跳动的次数。当隐藏或显示时，会添加半个跳动。	5

实例

使用跳动特效（Pulsate Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>跳动特效（Pulsate Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "pulsate" );
});
</script>

</body>
</html>
```

jQuery UI API - .removeClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。

返回：jQuery

```
.removeClass( className [, duration ] [, easing ] [, complete ] )
```

参数	类型	描述	默认值
className	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.removeClass( className [, options ] )
```

参数	类型	描述
className	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值：`swing`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。

- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.removeClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画的结尾改变。

该插件扩展了 jQuery 内置的 .removeClass() 方法。如果 jQuery UI 未加载，调用 `.removeClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

从匹配的元素中移除 class "big-blue" 。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.removeClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div class="big-blue"></div>

<script>
$( "div" ).click(function() {
  $( this ).removeClass( "big-blue", 1000, "easeInBack" );
});
</script>

</body>
</html>
```

jQuery UI API - 缩放特效（Scale Effect）

所属类别

特效（Effects）

用法

描述：按照某个百分比缩放元素。

```
scale
```

参数	类型	描述	默认值
direction	String	特效的方向。可能的值："both"、"vertical" 或 "horizontal"。	"both"
origin	Array	消失点。	["middle", "center"]
percent	Number	要缩放到的百分比。	
scale	String	元素的哪个区域将被调整尺寸："both"、"box"、"content"。当值为 "box" 时，调整元素的边框（border）和内边距（padding）的尺寸。当值为 "content" 时，调整元素内的所有内容的尺寸。	"both"

实例

实例 **1**：

使用缩放特效（Scale Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>缩放特效（Scale Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "scale" );
});
</script>

</body>
</html>
```

实例 **2**：

只在一个方向上使用缩放特效（Scale Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>>缩放特效（Scale Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle({ effect: "scale", direction: "horizontal"
});
</script>

</body>
</html>
```

jQuery UI API - 震动特效（Shake Effect）

所属类别

特效（Effects）

用法

描述：垂直或水平方向多次震动元素。

```
shake
```

参数	类型	描述	默认值
direction	String	"left" 或 "right" 的值将水平震动元素，"up" 或 "down" 的值将垂直震动元素。该值指定元素沿轴线移动时第一步的方向。	"left"
distance	Number	要震动的距离。	20
times	Integer	要震动的次数。	3

实例

震动一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>震动特效（Shake Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行震动。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).effect( "shake" );
});
</script>

</body>
</html>
```

jQuery UI API - .show()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Methods）

用法

描述：使用自定义效果来显示匹配的元素。

返回：jQuery

```
.show( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.show( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串

表示的队列中。

该插件扩展自 jQuery 内置的 .show() 方法。如果 jQuery UI 未加载，调用 `.show()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

使用折叠特效（Fold Effect）显示一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.show() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    display: none;
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>显示 div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).show( "fold", 1000 );
});
</script>

</body>
</html>
```

jQuery UI API - 尺寸特效（Size Effect）

所属类别

特效（Effects）

用法

描述：调整元素尺寸到指定宽度和高度。

```
size
```

参数	类型	描述	默认值
to	Object	要调整到的高度和宽度。	
origin	Array	消失点。	["top", "left"]
scale	String	元素的哪个区域将被调整尺寸："both"、"box"、"content"。当值为 "box" 时，调整元素的边框（border）和内边距（padding）的尺寸。当值为 "content" 时，调整元素内的所有内容的尺寸。	"both"

实例

使用尺寸特效（Size Effect）调整元素尺寸。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>尺寸特效（Size Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行尺寸调整。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).effect( "size", {
    to: { width: 200, height: 60 }
  }, 1000 );
});
</script>

</body>
</html>
```

jQuery UI API - 滑动特效（Slide Effect）

所属类别

特效（Effects）

用法

描述：把元素滑动出视区。

```
slide
```

参数	类型	描述	默认值
direction	String	特效的方向。可能的值："left"、"right"、"up"、"down"。	"both"
distance	Number	特效的距离。默认为元素的高度（height）还是宽度（width）取决于 direction 参数。可以设置为小于元素的宽度（width）/高度（height）的任意整数。	元素的 outerWidth

实例

使用滑动特效（Slide Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>滑动特效（Slide Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #toggle {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>点击任意地方进行切换。</p>
<div id="toggle"></div>

<script>
$( document ).click(function() {
  $( "#toggle" ).toggle( "slide" );
});
</script>

</body>
</html>
```

jQuery UI API - .switchClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。

返回：jQuery

```
.switchClass( removeClassName, addClassName [, duration ] [, easing
```

参数	类型	描述	默认值
removeClassName	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔。	
addClassName	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.switchClass( removeClassName, addClassName [, options ] )
```

参数	类型	描述
removeClassName	String	要从每个匹配的元素的 class 属性中移除的一个或多个 class 名称，多个 class 名称用空格分隔 。
addClassName	String	要添加到每个匹配的元素的 class 属性中的一个或多个 class 名称，多个 class 名称用空格分隔。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值： `400` ） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值： `swing` ） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串表示的队列中。

`.switchClass()` 方法允许您在动画过渡中，添加 Class 的同时移除 Class。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的，而不需要使用 JavaScript。所有的 class 动画方法，包括 `.switchClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画结束时改变。

该插件扩展自 jQuery 内置的 .switchClass() 方法。如果 jQuery UI 未加载，调用 `.switchClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

添加 class "blue" 到匹配的元素，并从匹配的元素中移除 class "big"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.switchClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big {
    width: 200px;
    height: 200px;
  }
  .blue {
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div class="big"></div>

<script>
$( "div" ).click(function() {
  $( this ).switchClass( "big", "blue", 1000, "easeInOutQuad" );
});
</script>

</body>
</html>
```

jQuery UI API - .toggle()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides） | 方法（Methods）

用法

描述：使用自定义效果来显示或隐藏匹配的元素。

返回：jQuery

```
.toggle( effect [, options ] [, duration ] [, complete ] )
```

参数	类型	描述	默认值
effect	String	一个字符串，指示要使用哪一种特效。	
options	Object	特效具体的设置和 easing。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
complete	Function()	一旦动画完成时要调用的函数。	

```
.toggle( options )
```

参数	类型	描述
options	Object	所有的动画设置。唯一一个必需的属性是 effect。

- **effect** 类型：String 描述：一个字符串，指示要使用哪一种特效。
- **easing**（默认值：`"swing"`） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **duration**（默认值：`400`） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **queue**（默认值：`true`） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false，动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串，在这种情况下，动画添加到由字符串

表示的队列中。

该插件扩展自 jQuery 内置的 .toggle() 方法。如果 jQuery UI 未加载，调用 `.toggle()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

使用折叠特效（Fold Effect）切换一个 div。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.toggle() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background: #ccc;
    border: 1px solid #000;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>切换 div</button>
<div></div>

<script>
$( "button" ).click(function() {
  $( "div" ).toggle( "fold", 1000 );
});
</script>

</body>
</html>
```

jQuery UI API - .toggleClass()

所属类别

特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）

用法

描述：当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。

返回：jQuery

```
.toggleClass( className [, switch ] [, duration ] [, easing ] [, co
```

参数	类型	描述	默认值
ClassName	String	为匹配的元素集合中的每个元素要切换的一个或多个 class 名称，多个 class 名称用空格分隔 。	
switch	Boolean	一个布尔值，指定 class 应被添加还是被移除。	
duration	Number 或 String	一个字符串或一个数字，指定动画将运行多久。	400
easing	String	一个字符串，指示要使用的 easing 函数。	swing
complete	Function()	一旦动画完成时要调用的函数。	

```
.toggleClass( className [, switch ] [, options ] )
```

参数	类型	描述
ClassName	String	为匹配的元素集合中的每个元素要切换的一个或多个 class 名称， 多个 class 名称用空格分隔 。
switch	Boolean	一个布尔值，指定 class 应被添加还是被移除。
options	Object	所有的动画设置。所有的属性是可选的。

- **duration**（默认值： `400` ） 类型：Number 或 String 描述：一个字符串或一个数字，指定动画将运行多久。
- **easing**（默认值： `swing` ） 类型：String 描述：一个字符串，指示要使用的 easing 函数。
- **complete** 类型：Function() 描述：一旦动画完成时要调用的函数。
- **children**（默认值： `false` ） 类型：Boolean 描述：指定动画是否被应用到匹配元素的所有后代。此功能应慎重使用，因为判断元素是否是动画的后代的代价是很大的，会根据后代的数量线性增长。
- **queue**（默认值： `true` ） 类型：Boolean 或 String 描述：一个布尔值，指示是否将动画放在特效队列中。如果是 false， 动画将立即开始。自 **jQuery 1.7** 起，queue 选项也接受一个字符串， 在这种情况下， 动画添加到由字符串表示的队列中。

与原生的 CSS 过渡相似，jQuery UI 的 class 动画提供了一个平稳的从一个状态转换到另一个状态的过渡，同时确保所有样式变化的细节是通过 CSS 来完成的， 而不需要使用 JavaScript。所有的 class 动画方法，包括 `.toggleClass()` ，允许自定义动画持续时间和 easing 函数，在动画完成时也提供了一个回调。

并非所有的样式都可以进行动画添加。例如，对背景图像进行动画化。任何不能动画化的样式都将在动画结束时改变。

该插件扩展自 jQuery 内置的 .toggleClass() 方法。如果 jQuery UI 未加载，调用 `.toggleClass()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

为匹配的元素切换 class "big-blue"。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.toggleClass() 演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
  .big-blue {
    width: 200px;
    height: 200px;
    background-color: #00f;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div></div>

<script>
$( "div" ).click(function() {
  $( this ).toggleClass( "big-blue", 1000, "easeOutSine" );
});
</script>

</body>
</html>
```

jQuery UI API - 转移特效（Transfer Effect）

所属类别

特效（Effects）

用法

描述：把一个元素的轮廓转移到另一个元素。

```
transfer
```

参数	类型	描述
className	String	转移的元素将接收的参数化的 class 名称。
to	String	jQuery 选择器，要转移到的元素。

当尝试两个元素之间的可视化交互时非常有用。

转移的元素本身带有 class `ui-effects-transfer` ，其他的样式需要由您来定义，比如添加背景或边框。

实例

在绿色元素上点击把它转移到另一个元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>转移特效（Transfer Effect）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div.green {
    width: 100px;
    height: 80px;
    background: green;
    border: 1px solid black;
    position: relative;
  }
  div.red {
    margin-top: 10px;
    width: 50px;
    height: 30px;
    background: red;
    border: 1px solid black;
    position: relative;
  }
  .ui-effects-transfer {
    border: 1px dotted black;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div class="green"></div>
<div class="red"></div>

<script>
$( "div" ).click(function() {
  var i = 1 - $( "div" ).index( this );
  $( this ).effect( "transfer", { to: $( "div" ).eq( i ) }, 1000 );
});
</script>

</body>
</html>
```

jQuery UI API 类别 - 特效核心（Effects Core）

由 jquery.ui.effect.js 提供的功能。除了下面列出的方法，jquery.ui.effect.js 还包括一些 Easings。

API	描述	也属于类别
.addClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。	特效（Effects） | 方法重载（Method Overrides）
颜色动画（Color Animation）	使用 .animate() 实现颜色动画效果。	
.effect()	对一个元素应用动画特效。	特效（Effects） | 方法（Method）
.hide()	使用自定义效果来隐藏匹配的元素。	特效（Effects） | 方法重载（Method Overrides） | 方法（Method）
.removeClass()	当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。	特效（Effects） | 方法重载（Method Overrides）
.show()	使用自定义效果来显示匹配的元素。	特效（Effects） | 方法重载（Method Overrides） | 方法（Method）
.switchClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加和移除指定的 Class。	特效（Effects）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效（Effects） | 方法重载（Method Overrides） | 方法（Method）
.toggleClass()	当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。	特效（Effects） | 方法重载（Method Overrides）

jQuery UI API - 颜色动画（Color Animation）

jQuery UI 特效核心添加了使用 `rgb()` 、 `rgba()` 、十六进制值或者颜色名比如 `"aqua"` 来动态改变 color 属性的功能。只需要包含 jQuery UI 特效核心文件， `.animate()` 就会支持颜色。

支持下列属性：

- `backgroundColor`
- `borderBottomColor`
- `borderLeftColor`
- `borderRightColor`
- `borderTopColor`
- `color`
- `columnRuleColor`
- `outlineColor`
- `textDecorationColor`
- `textEmphasisColor`

对颜色动画的支持来自 jQuery Color 插件。Color 插件提供了一些用于颜色的函数。如需查看完整文档，请访问 jQuery Color 文档。

Class 动画（Class Animations）

虽然可以直接对 color 属性进行动画化，但是通常采用另一种更好的方法，即在一个 class 中包含样式。jQuery UI 提供了一些动态添加或去除 CSS 类的方法，分别是 `.addClass()` 、 `.removeClass()` 、 `.toggleClass()` 和 `.switchClass()` 。这些方法将自动确定哪些属性需要改变，哪些需要应用适当的动画。

实例

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>颜色动画（Color Animation）演示</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/the
  <style>
  #elem {
    color: #006;
    background-color: #aaa;
    font-size: 25px;
    padding: 1em;
    text-align: center;
  }
  </style>
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></sc
</head>
<body>

<div id="elem">颜色动画</div>
<button id="toggle">改变颜色</button>

<script>
$( "#toggle" ).click(function() {
  $( "#elem" ).animate({
    color: "green",
    backgroundColor: "rgb( 20, 20, 20 )"
  });
});
</script>

</body>
</html>
```

jQuery UI API 类别 - 交互（Interactions）

jQuery UI 提供了一套基于鼠标的交互。

API	描述	也属于类别
可拖拽小部件（Draggable Widget）	允许使用鼠标移动元素。	
可放置小部件（Droppable Widget）	为可拖拽小部件创建目标。	
鼠标交互（Mouse Interaction）	基本交互层。	实用工具（Utilities）
可调整尺寸小部件（Resizable Widget）	使用鼠标改变元素的尺寸。	
可选择小部件（Selectable Widget）	使用鼠标选择单个元素或一组元素。	
可排序小部件（Sortable Widget）	使用鼠标调整列表中或者网格中元素的排序。	

jQuery UI API - 可拖拽小部件（Draggable Widget）

所属类别

交互（Interactions）

用法

描述：允许使用鼠标移动元素。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：让被选元素可被鼠标拖拽。如果您不只是拖拽，而是拖拽 & 放置，请查看 jQuery UI 可放置（Droppable）插件，为可拖拽元素提供了一个放置目标。

快速导航

addClassesappendToaxiscancelconnectToSortablecontainmentcursorcursorAtde

addClasses

类型：Boolean

描述：如果设置为 `false`，将阻止 `ui-draggable` class 被添加。当在数百个元素上调用 `.draggable()` 时，这么设置有利于性能优化。

代码实例：

初始化带有指定 `addClasses` 选项的 draggable：

```
$( ".selector" ).draggable({ addClasses: false });
```

在初始化后，获取或设置 `addClasses` 选项：

```
// getter
var addClasses = $( ".selector" ).draggable( "option", "addClasses'

// setter
$( ".selector" ).draggable( "option", "addClasses", false );
```

默认值：true

appendTo

类型：jQuery 或 Element 或 Selector 或 String

描述：当拖拽时，draggable 助手（helper）要追加到哪一个元素。

支持多个类型：

- **jQuery**：一个 jQuery 对象，包含助手（helper）要追加到的元素。
- **Element**：要追加助手（helper）的元素。
- **Selector**：一个选择器，指定哪一个元素要追加助手（helper）。
- **String**：字符串 `"parent"` 将促使助手（helper）成为 draggable 的同级。

代码实例：

初始化带有指定 `appendTo` 选项的 draggable：

```
$( ".selector" ).draggable({ appendTo: "body" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).draggable( "option", "appendTo" );

// setter
$( ".selector" ).draggable( "option", "appendTo", "body" );
```

默认值："parent"

axis

类型：String

描述：约束在水平轴 (x) 或垂直轴 (y) 上拖拽。可能的值：`"x"`，`"y"`。

代码实例：

初始化带有指定 `axis` 选项的 draggable：

```
$( ".selector" ).draggable({ axis: "x" });
```

在初始化后，获取或设置 `axis` 选项：

```
// getter
var axis = $( ".selector" ).draggable( "option", "axis" );

// setter
$( ".selector" ).draggable( "option", "axis", "x" );
```

默认值：false

cancel

类型：Selector

描述：防止从指定的元素上开始拖拽。

代码实例：

初始化带有指定 `cancel` 选项的 draggable：

```
$( ".selector" ).draggable({ cancel: ".title" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).draggable( "option", "cancel" );

// setter
$( ".selector" ).draggable( "option", "cancel", ".title" );
```

默认值："input, textarea, button, select, option"

connectToSortable

类型：Selector

描述：允许 draggable 放置在指定的 sortable 上。如果使用了该选项，一个 draggable 可被放置在一个 sortable 列表上，然后成为列表的一部分。注意： `helper` 选项必须设置为 `"clone"` ，以便更好地工作。必须包含 可排序小部件（Sortable Widget）。

代码实例：

初始化带有指定 `connectToSortable` 选项的 draggable：

```
$( ".selector" ).draggable({ connectToSortable: "#my-sortable" });
```

在初始化后，获取或设置 `connectToSortable` 选项：

```
// getter
var connectToSortable = $( ".selector" ).draggable( "option", "conn

// setter
$( ".selector" ).draggable( "option", "connectToSortable", "#my-sor
```

默认值：false

containment

类型：Selector 或 Element 或 String 或 Array

描述：约束在指定元素或区域的边界内拖拽。

支持多个类型：

- **Selector**：可拖拽元素将被包含在 selector 第一个元素的边界内。如果未找到元素，则不设置 containment。
- **Element**：可拖拽元素将被韩寒在元素的边界。
- **String**：可能的值：`"parent"`、`"document"`、`"window"`。
- **Array**：一个数组，以形式 `[ x1, y1, x2, y2 ]` 定义元素的边界。

代码实例：

初始化带有指定 `containment` 选项的 draggable：

```
$( ".selector" ).draggable({ containment: "parent" });
```

在初始化后，获取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).draggable( "option", "containmen

// setter
$( ".selector" ).draggable( "option", "containment", "parent" );
```

默认值：false

cursor

类型：String

描述：拖拽操作期间的 CSS 光标。

代码实例：

初始化带有指定 `cursor` 选项的 draggable：

```
$( ".selector" ).draggable({ cursor: "crosshair" });
```

在初始化后，获取或设置 `cursor` 选项：

```
// getter
var cursor = $( ".selector" ).draggable( "option", "cursor" );

// setter
$( ".selector" ).draggable( "option", "cursor", "crosshair" );
```

默认值："auto"

cursorAt

类型：Object

描述：设置拖拽助手（helper）相对于鼠标光标的偏移。坐标可通过一个或两个键的组合成一个哈希给出：`{ top, left, right, bottom }`。

代码实例：

初始化带有指定 `cursorAt` 选项的 draggable：

```
$( ".selector" ).draggable({ cursorAt: { left: 5 } });
```

在初始化后，获取或设置 `cursorAt` 选项：

```
// getter
var cursorAt = $( ".selector" ).draggable( "option", "cursorAt" );

// setter
$( ".selector" ).draggable( "option", "cursorAt", { left: 5 } );
```

默认值：false

delay

类型：Number

描述：鼠标按下后直到拖拽开始为止的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的拖拽。

代码实例：

初始化带有指定 `delay` 选项的 draggable：

```
$( ".selector" ).draggable({ delay: 300 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).draggable( "option", "delay" );

// setter
$( ".selector" ).draggable( "option", "delay", 300 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 draggable。

代码实例：

初始化带有指定 `disabled` 选项的 draggable：

```
$( ".selector" ).draggable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).draggable( "option", "disabled" );

// setter
$( ".selector" ).draggable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后拖拽开始前必须移动的距离，以像素计。该选项可以防止点击在某个元素上时不必要的拖拽。

代码实例：

初始化带有指定 `distance` 选项的 draggable：

```
$( ".selector" ).draggable({ distance: 10 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).draggable( "option", "distance" );

// setter
$( ".selector" ).draggable( "option", "distance", 10 );
```

默认值：1

grid

类型：Array

描述：对齐拖拽助手（helper）到网格，每个 x 和 y 像素。数组形式必须是 `[ x, y ]`。

代码实例：

初始化带有指定 `grid` 选项的 draggable：

```
$( ".selector" ).draggable({ grid: [ 50, 20 ] });
```

在初始化后，获取或设置 `grid` 选项：

```
// getter
var grid = $( ".selector" ).draggable( "option", "grid" );

// setter
$( ".selector" ).draggable( "option", "grid", [ 50, 20 ] );
```

默认值：false

handle

类型：Selector 或 Element

描述：如果指定了该选项，则限制开始拖拽，除非鼠标在指定的元素上按下。只有可拖拽（draggable）元素的后代元素才允许被拖拽。

代码实例：

初始化带有指定 `handle` 选项的 draggable：

```
$( ".selector" ).draggable({ handle: "h2" });
```

在初始化后，获取或设置 `handle` 选项：

```
// getter
var handle = $( ".selector" ).draggable( "option", "handle" );

// setter
$( ".selector" ).draggable( "option", "handle", "h2" );
```

默认值：false

helper

类型：String 或 Function()

描述：允许一个 helper 元素用于拖拽显示。

支持多个类型：

- **String**：如果设置为 `"clone"`，元素将被克隆，且克隆将被拖拽。
- **Function**：一个函数，将返回拖拽时要使用的 DOMElement。

代码实例：

初始化带有指定 `helper` 选项的 draggable：

```
$( ".selector" ).draggable({ helper: "clone" });
```

在初始化后，获取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).draggable( "option", "helper" );

// setter
$( ".selector" ).draggable( "option", "helper", "clone" );
```

默认值："original"

iframeFix

类型：Boolean 或 Selector

描述：防止拖拽期间 iframes 捕捉鼠标移动（mousemove ）事件。在与 `cursorAt` 选项结合使用时，或鼠标光标未覆盖在助手（helper）上时，非常有用。

支持多个类型：

- **Boolean**：当设置为 `true` 时，透明遮罩将被放置在页面上所有 iframes 上。
- **Selector**：匹配 selector 的任意 iframes 将被透明遮罩覆盖。

代码实例：

初始化带有指定 `iframeFix` 选项的 draggable：

```
$( ".selector" ).draggable({ iframeFix: true });
```

在初始化后，获取或设置 `iframeFix` 选项：

```
// getter
var iframeFix = $( ".selector" ).draggable( "option", "iframeFix" )

// setter
$( ".selector" ).draggable( "option", "iframeFix", true );
```

默认值：false

opacity

类型：Number

描述：当被拖拽时助手（helper）的不透明度。

代码实例：

初始化带有指定 `opacity` 选项的 draggable：

```
$( ".selector" ).draggable({ opacity: 0.35 });
```

在初始化后，获取或设置 `opacity` 选项：

```
// getter
var opacity = $( ".selector" ).draggable( "option", "opacity" );

// setter
$( ".selector" ).draggable( "option", "opacity", 0.35 );
```

默认值：false

refreshPositions

类型：Boolean

描述：如果设置为 `true` ，在每次鼠标移动（mousemove）时都会计算所有可放置的位置。注意：这解决了高度动态的问题，但是明显降低了性能。

代码实例：

初始化带有指定 `refreshPositions` 选项的 draggable：

```
$( ".selector" ).draggable({ refreshPositions: true });
```

在初始化后，获取或设置 `refreshPositions` 选项：

```
// getter
var refreshPositions = $( ".selector" ).draggable( "option", "refre

// setter
$( ".selector" ).draggable( "option", "refreshPositions", true );
```

默认值：false

revert

类型：Boolean 或 String 或 Function()

描述：当拖拽停止时，元素是否还原到它的开始位置。

支持多个类型：

- **Boolean**：如果设置为 `true`，元素总会还原。
- **String**：如果设置为 `"invalid"`，还原仅在 draggable 未放置在 droppable 上时发生，如果设置为 `"valid"` 则相反。
- **Function**：一个函数，确定元素是否还原到它的开始位置。该函数必须返回 `true` 才能还原元素。

代码实例：

初始化带有指定 `revert` 选项的 draggable：

```
$( ".selector" ).draggable({ revert: true });
```

在初始化后，获取或设置 `revert` 选项：

```
// getter
var revert = $( ".selector" ).draggable( "option", "revert" );

// setter
$( ".selector" ).draggable( "option", "revert", true );
```

默认值：false

revertDuration

类型：Number

描述：还原（revert）动画的持续时间，以毫秒计。如果 `revert` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `revertDuration` 选项的 draggable：

```
$( ".selector" ).draggable({ revertDuration: 200 });
```

在初始化后，获取或设置 `revertDuration` 选项：

```
// getter
var revertDuration = $( ".selector" ).draggable( "option", "revertD

// setter
$( ".selector" ).draggable( "option", "revertDuration", 200 );
```

默认值：500

scope

类型：String

描述：用于组合配套 draggable 和 droppable 项，除了 droppable 的 `accept` 选项之外。一个与 droppable 带有相同的 `scope` 值的 draggable 会被该 droppable 接受。

代码实例：

初始化带有指定 `scope` 选项的 draggable：

```
$( ".selector" ).draggable({ scope: "tasks" });
```

在初始化后，获取或设置 `scope` 选项：

```
// getter
var scope = $( ".selector" ).draggable( "option", "scope" );

// setter
$( ".selector" ).draggable( "option", "scope", "tasks" );
```

默认值："default"

scroll

类型：Boolean

描述：如果设置为 `true`，当拖拽时容器会自动滚动。

代码实例：

初始化带有指定 `scroll` 选项的 draggable：

```
$( ".selector" ).draggable({ scroll: false });
```

在初始化后，获取或设置 `scroll` 选项：

```
// getter
var scroll = $( ".selector" ).draggable( "option", "scroll" );

// setter
$( ".selector" ).draggable( "option", "scroll", false );
```

默认值：true

scrollSensitivity

类型：Number

描述：从要滚动的视区边缘起的距离，以像素计。距离是相对于指针的，不是相对于 draggable。如果 `scroll` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `scrollSensitivity` 选项的 draggable：

```
$( ".selector" ).draggable({ scrollSensitivity: 100 });
```

在初始化后，获取或设置 `scrollSensitivity` 选项：

```
// getter
var scrollSensitivity = $( ".selector" ).draggable( "option", "scro

// setter
$( ".selector" ).draggable( "option", "scrollSensitivity", 100 );
```

默认值：20

scrollSpeed

类型：Number

描述：当鼠标指针获取到在 `scrollSensitivity` 距离内时，窗体滚动的速度。如果 `scroll` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `scrollSpeed` 选项的 draggable：

```
$( ".selector" ).draggable({ scrollSpeed: 100 });
```

在初始化后，获取或设置 `scrollSpeed` 选项：

```
// getter
var scrollSpeed = $( ".selector" ).draggable( "option", "scrollSpee

// setter
$( ".selector" ).draggable( "option", "scrollSpeed", 100 );
```

默认值：20

snap

类型：Boolean 或 Selector

描述：元素是否对齐到其他元素。

支持多个类型：

- **Boolean**：当设置为 `true` 时，元素会对齐到其它可拖拽（draggable）元素。
- **Selector**：一个选择器，指定要对齐到哪个元素。

代码实例：

初始化带有指定 `snap` 选项的 draggable：

```
$( ".selector" ).draggable({ snap: true });
```

在初始化后，获取或设置 `snap` 选项：

```
// getter
var snap = $( ".selector" ).draggable( "option", "snap" );

// setter
$( ".selector" ).draggable( "option", "snap", true );
```

默认值：false

snapMode

类型：String

描述：决定 draggable 将对齐到对齐元素的哪个边缘。如果 `snap` 选项是 `false` 则忽略。可能的值：`"inner"`、`"outer"`、`"both"`。

代码实例：

初始化带有指定 `snapMode` 选项的 draggable：

```
$( ".selector" ).draggable({ snapMode: "inner" });
```

在初始化后，获取或设置 `snapMode` 选项：

```
// getter
var snapMode = $( ".selector" ).draggable( "option", "snapMode" );

// setter
$( ".selector" ).draggable( "option", "snapMode", "inner" );
```

默认值："both"

snapTolerance

类型：Number

描述：从要发生对齐的对齐元素边缘起的距离，以像素计。如果 `snap` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `snapTolerance` 选项的 draggable：

```
$( ".selector" ).draggable({ snapTolerance: 30 });
```

在初始化后，获取或设置 `snapTolerance` 选项：

```
// getter
var snapTolerance = $( ".selector" ).draggable( "option", "snapTole

// setter
$( ".selector" ).draggable( "option", "snapTolerance", 30 );
```

默认值：20

stack

类型：Selector

描述：控制匹配选择器（selector）的元素集合的 z-index，总是在当前拖拽项的前面，在类似窗口管理器这样的事物中非常有用。

代码实例：

初始化带有指定 `stack` 选项的 draggable：

```
$( ".selector" ).draggable({ stack: ".products" });
```

在初始化后，获取或设置 `stack` 选项：

```
// getter
var stack = $( ".selector" ).draggable( "option", "stack" );

// setter
$( ".selector" ).draggable( "option", "stack", ".products" );
```

默认值：false

zIndex

类型：Number

描述：当被拖拽时，助手（helper）的 Z-index。

代码实例：

初始化带有指定 `zIndex` 选项的 draggable：

```
$( ".selector" ).draggable({ zIndex: 100 });
```

在初始化后，获取或设置 `zIndex` 选项：

```
// getter
var zIndex = $( ".selector" ).draggable( "option", "zIndex" );

// setter
$( ".selector" ).draggable( "option", "zIndex", 100 );
```

默认值：false

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 draggable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).draggable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 draggable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).draggable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 draggable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).draggable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).draggable( "option", "disabled" 
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 draggable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).draggable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 draggable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).draggable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 draggable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).draggable( "option", { disabled: true } );
```

widget()

类型：jQuery

描述：返回一个包含 draggable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).draggable( "widget" );
```

事件

create(event, ui)

类型：dragcreate

描述：当 draggable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 draggable：

```
$( ".selector" ).draggable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 dragcreate 事件：

```
$( ".selector" ).on( "dragcreate", function( event, ui ) {} );
```

drag(event, ui)

类型：drag

描述：在拖拽期间当鼠标移动时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
    - **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
    - **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 drag 回调的 draggable：

```
$( ".selector" ).draggable({
  drag: function( event, ui ) {}
});
```

绑定一个事件监听器到 drag 事件：

```
$( ".selector" ).on( "drag", function( event, ui ) {} );
```

start(event, ui)

类型：dragstart

描述：当拖拽开始时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
    - **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
    - **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 start 回调的 draggable：

```
$( ".selector" ).draggable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 dragstart 事件：

```
$( ".selector" ).on( "dragstart", function( event, ui ) {} );
```

stop(event, ui)

类型：dragstop

描述：当拖拽停止时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
    - **position** 类型：Object 描述：助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
    - **offset** 类型：Object 描述：助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 stop 回调的 draggable：

```
$( ".selector" ).draggable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 dragstop 事件：

```
$( ".selector" ).on( "dragstop", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 可拖拽小部件（Draggable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可拖拽小部件（Draggable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #draggable {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="draggable">请拖拽我！</div>

<script>
$( "#draggable" ).draggable();
</script>

</body>
</html>
```

jQuery UI API - 可放置小部件（Droppable Widget）

所属类别

交互（Interactions）

用法

描述：为可拖拽小部件创建目标。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可放置（Droppable）插件让被选元素可放置（意味着它们在拖拽后接受被放置）。您可以逐个指定哪一个 draggable 会接受。

快速导航

选项	
acceptactiveClassaddClassesdisabledgreedyhoverClassscopetolerance	destr

accept

类型：Selector 或 Function()

描述：控制哪个可拖拽（draggable）元素可被 droppable 接受。

支持多个类型：

- **Selector**：一个选择器，指定哪个可拖拽（draggable）元素可被 droppable 接受。
- **Function()**：一个函数，将被页面上每个 draggable 调用（作为第一个参数传递给函数）。如果 draggable 被接受，该函数必须返回 `true` 。

代码实例：

初始化带有指定 `accept` 选项的 droppable：

```
$( ".selector" ).droppable({ accept: ".special" });
```

在初始化后，获取或设置 `accept` 选项：

```
// getter
var accept = $( ".selector" ).droppable( "option", "accept" );

// setter
$( ".selector" ).droppable( "option", "accept", ".special" );
```

默认值："*"

activeClass

类型：String

描述：如果指定了该选项，当一个可接受的 draggable 被拖拽时，class 将被添加到 droppable。

代码实例：

初始化带有指定 `activeClass` 选项的 droppable：

```
$( ".selector" ).droppable({ activeClass: "ui-state-highlight" });
```

在初始化后，获取或设置 `activeClass` 选项：

```
// getter
var activeClass = $( ".selector" ).droppable( "option", "activeClas

// setter
$( ".selector" ).droppable( "option", "activeClass", "ui-state-high
```

默认值：false

addClasses

类型：Boolean

描述：如果设置为 `false`，将防止 `ui-droppable` class 被添加。这在数百个元素上调用 `.droppable()` 时有助于性能优化。

代码实例：

初始化带有指定 `addClasses` 选项的 droppable：

```
$( ".selector" ).droppable({ addClasses: false });
```

在初始化后，获取或设置 `addClasses` 选项：

```
// getter
var addClasses = $( ".selector" ).droppable( "option", "addClasses"

// setter
$( ".selector" ).droppable( "option", "addClasses", false );
```

默认值：true

disabled

类型：Boolean

描述：如果设置为 `true` ，则禁用该 droppable。

代码实例：

初始化带有指定 `disabled` 选项的 droppable：

```
$( ".selector" ).droppable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).droppable( "option", "disabled" );

// setter
$( ".selector" ).droppable( "option", "disabled", true );
```

默认值：false

greedy

类型：Boolean

描述：默认情况下，当一个元素被放置在嵌套是 droppable 上时，每个 droppable 将接收该元素。然而，通过设置该选项为 `true` ，任何父元素的 droppable 将无法接收该元素。 `drop` 事件仍将照常，但会检查 `event.target` 以便查看哪个

droppable 接收 draggable 元素。

代码实例：

初始化带有指定 `greedy` 选项的 droppable：

```
$( ".selector" ).droppable({ greedy: true });
```

在初始化后，获取或设置 `greedy` 选项：

```
// getter
var greedy = $( ".selector" ).droppable( "option", "greedy" );

// setter
$( ".selector" ).droppable( "option", "greedy", true );
```

默认值：false

hoverClass

类型：String

描述：如果指定了该选项，当一个可接受 draggable 被覆盖在 droppable 上时，class 将被添加到 droppable。

代码实例：

初始化带有指定 `hoverClass` 选项的 droppable：

```
$( ".selector" ).droppable({ hoverClass: "drop-hover" });
```

在初始化后，获取或设置 `hoverClass` 选项：

```
// getter
var hoverClass = $( ".selector" ).droppable( "option", "hoverClass'

// setter
$( ".selector" ).droppable( "option", "hoverClass", "drop-hover" );
```

默认值：false

scope

类型：String

描述：用于组合配套 draggable 和 droppable 项，除了 droppable 的 `accept` 选项之外。一个与 droppable 带有相同的 `scope` 值的 draggable 会被该 droppable 接受。

代码实例：

初始化带有指定 `scope` 选项的 droppable：

```
$( ".selector" ).droppable({ scope: "tasks" });
```

在初始化后，获取或设置 `scope` 选项：

```
// getter
var scope = $( ".selector" ).droppable( "option", "scope" );

// setter
$( ".selector" ).droppable( "option", "scope", "tasks" );
```

默认值："default"

tolerance

类型：String

描述：指定用于测试一个 draggable 是否覆盖在一个 droppable 上的模式。可能的值：

- `"fit"`：draggable 完全重叠在 droppable 上。
- `"intersect"`：draggable 重叠在 droppable 上，两个方向上至少 50%。
- `"pointer"`：鼠标指针重叠在 droppable 上。
- `"touch"`：draggable 重叠在 droppable 上，任何数量皆可。

代码实例：

初始化带有指定 `tolerance` 选项的 droppable：

```
$( ".selector" ).droppable({ tolerance: "fit" });
```

在初始化后，获取或设置 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).droppable( "option", "tolerance" 

// setter
$( ".selector" ).droppable( "option", "tolerance", "fit" );
```

默认值："intersect"

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 droppable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).droppable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 droppable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).droppable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 droppable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).droppable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).droppable( "option", "disabled" )
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 droppable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).droppable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 droppable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).droppable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 droppable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).droppable( "option", { disabled: true } );
```

widget()

类型：jQuery

描述：返回一个包含 droppable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).droppable( "widget" );
```

事件

activate(event, ui)

类型：dropactivate

描述：当一个可接受的 draggable 开始拖拽时触发。如果您想让 droppable 被放置时"点亮"，该选项就可以派上用场。

- **event** 类型：Event
- **ui** 类型：Object
    - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
    - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
    - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 activate 回调的 droppable：

```
$( ".selector" ).droppable({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropactivate 事件：

```
$( ".selector" ).on( "dropactivate", function( event, ui ) {} );
```

create(event, ui)

类型：dropcreate

描述：当 droppable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 droppable：

```
$( ".selector" ).droppable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropcreate 事件：

```
$( ".selector" ).on( "dropcreate", function( event, ui ) {} );
```

deactivate(event, ui)

类型：dropdeactivate

描述：当一个可接受的 draggable 停止拖拽时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，

比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 deactivate 回调的 droppable：

```
$( ".selector" ).droppable({
  deactivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropdeactivate 事件：

```
$( ".selector" ).on( "dropdeactivate", function( event, ui ) {} );
```

drop(event, ui)

类型：drop

描述：当一个可接受的 draggable 被放置在 droppable（基于 `[tolerance](#option-tolerance)` 选项）上时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 drop 回调的 droppable：

```
$( ".selector" ).droppable({
  drop: function( event, ui ) {}
});
```

绑定一个事件监听器到 drop 事件：

```
$( ".selector" ).on( "drop", function( event, ui ) {} );
```

out(event, ui)

类型：dropout

描述：当 droppable 被拖拽出 droppable（基于 `[tolerance](#option-tolerance)` 选项）时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 out 回调的 droppable：

```
$( ".selector" ).droppable({
  out: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropout 事件：

```
$( ".selector" ).on( "dropout", function( event, ui ) {} );
```

over(event, ui)

类型：dropover

描述：当一个可接受的 draggable 被拖拽在 droppable（基于 `[tolerance](#option-tolerance)` 选项）上时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **draggable** 类型：jQuery 描述：jQuery 对象，表示 draggable 元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被拖拽的助手（helper）。
  - **position** 类型：Object 描述：draggable 助手（helper）的当前 CSS 位置，比如 `{ top, left }` 对象。
  - **offset** 类型：Object 描述：draggable 助手（helper）的当前偏移位置，比如 `{ top, left }` 对象。

代码实例：

初始化带有指定 over 回调的 droppable：

```
$( ".selector" ).droppable({
  over: function( event, ui ) {}
});
```

绑定一个事件监听器到 dropover 事件：

```
$( ".selector" ).on( "dropover", function( event, ui ) {} );
```

实例

一对 draggable 和 droppable 元素。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可放置小部件（Droppable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #draggable {
    width: 100px;
    height: 100px;
    background: #ccc;
  }
  #droppable {
    position: absolute;
    left: 250px;
    top: 0;
    width: 125px;
    height: 125px;
    background: #999;
    color: #fff;
    padding: 10px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="droppable">请放置到这里！</div>
<div id="draggable">请拖拽我！</div>

<script>
$( "#draggable" ).draggable();
$( "#droppable" ).droppable({
  drop: function() {
    alert( "dropped" );
  }
});
</script>

</body>
</html>
```

jQuery UI API - 鼠标交互（Mouse Interaction）

所属类别

交互（Interactions） | 实用工具（Utilities）

用法

描述：基本交互层。

依赖：

- 部件库（Widget Factory）

注释：与 `jQuery.Widget` 相似，鼠标交互的目的不是直接使用。这是一个纯粹给其他小部件继承用的基础层。该页面有添加到 `jQuery.Widget` 的文档，但是它包含了不能被覆盖的内部方法。公共的 API 是 `_mouseStart()` 、 `_mouseDrag()` 、 `_mouseStop()` 和 `_mouseCapture()` 。

快速导航

选项	
canceldelaydistance	_mouseCapture_mouseDelayMet_mouseDestroy_mouse

cancel

类型：Selector

描述：防止从指定的元素上开始交互。

代码实例：

初始化带有指定 `cancel` 选项的 mouse：

```
$( ".selector" ).mouse({ cancel: ".title" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).mouse( "option", "cancel" );

// setter
$( ".selector" ).mouse( "option", "cancel", ".title" );
```

默认值："input, textarea, button, select, option"

delay

类型：Number

描述：鼠标按下后直至交互开始的事件，以毫秒计。该选项可用于防止点击在一个元素上时不必要的交互。

代码实例：

初始化带有指定 `delay` 选项的 mouse：

```
$( ".selector" ).mouse({ delay: 300 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).mouse( "option", "delay" );

// setter
$( ".selector" ).mouse( "option", "delay", 300 );
```

默认值：0

distance

类型：Number

描述：鼠标按下后交互开始前鼠标必须移动的距离，以像素计。该选项可用于防止点击在一个元素上时不必要的交互。

代码实例：

初始化带有指定 `distance` 选项的 mouse：

```
$( ".selector" ).mouse({ distance: 10 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).mouse( "option", "distance" );

// setter
$( ".selector" ).mouse( "option", "distance", 10 );
```

默认值：1

方法

_mouseCapture()

类型：Boolean

描述：决定交互是否应该基于交互的事件目标开始。默认的实现总是返回 `true` 。

- 该方法不接受任何参数。

代码实例：

调用 _mouseCapture 方法：

```
$( ".selector" ).mouse( "_mouseCapture" );
```

_mouseDelayMet()

类型：Boolean

描述：决定 `[delay](#option-delay)` 选项是否满足当前交互。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDelayMet 方法：

```
$( ".selector" ).mouse( "_mouseDelayMet" );
```

_mouseDestroy()

类型：jQuery (plugin only)

描述：销毁交互事件处理程序。这必须调用来自扩展的小部件的 `_destroy()` 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDestroy 方法：

```
$( ".selector" ).mouse( "_mouseDestroy" );
```

_mouseDistanceMet()

类型：Boolean

描述：决定 `[distance](#option-distance)` 选项是否满足当前交互。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDistanceMet 方法：

```
$( ".selector" ).mouse( "_mouseDistanceMet" );
```

_mouseDown()

类型：jQuery (plugin only)

描述：处理交互的开始。确认与主要的鼠标按钮关联的事件，确保 `[delay](#option-delay)` 与 `[distance](#option-distance)` 在交互启动之前得到满足。当交互已经准备开始，为要处理的扩展小部件调用 `[_mouseStart](#method-_mouseStart)` 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDown 方法：

```
$( ".selector" ).mouse( "_mouseDown" );
```

_mouseDrag()

类型：jQuery (plugin only)

描述：扩展小部件应实现一个 `_mouseDrag()` 方法，来处理交互的每个移动。该方法将接收与鼠标移动相关联的鼠标事件。

- 该方法不接受任何参数。

代码实例：

调用 _mouseDrag 方法：

```
$( ".selector" ).mouse( "_mouseDrag" );
```

_mouseInit()

类型：jQuery (plugin only)

描述：初始化交互事件处理程序。这必须调用来自扩展的小部件的 `_create()` 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseInit 方法：

```
$( ".selector" ).mouse( "_mouseInit" );
```

_mouseMove()

类型：jQuery (plugin only)

描述：处理交互的每个移动。为要处理的扩展小部件调用 _mouseDrag 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseMove 方法：

```
$( ".selector" ).mouse( "_mouseMove" );
```

_mouseStart()

类型：jQuery (plugin only)

描述：扩展小部件应实现一个 `_mouseStart()` 方法，来处理交互的开始。该方法将接收与交互开始相关联的鼠标事件。

- 该方法不接受任何参数。

代码实例：

调用 _mouseStart 方法：

```
$( ".selector" ).mouse( "_mouseStart" );
```

_mouseStop()

类型：jQuery (plugin only)

描述：扩展小部件应实现一个 `_mouseStop()` 方法，来处理交互的结束。该方法将接收与交互结束相关联的鼠标事件。

- 该方法不接受任何参数。

代码实例：

调用 _mouseStop 方法：

```
$( ".selector" ).mouse( "_mouseStop" );
```

_mouseUp()

类型：jQuery (plugin only)

描述：处理交互的结束。为要处理的扩展小部件调用 _mouseStop 方法。

- 该方法不接受任何参数。

代码实例：

调用 _mouseUp 方法：

```
$( ".selector" ).mouse( "_mouseUp" );
```

jQuery UI API - 可调整尺寸小部件（Resizable Widget）

所属类别

[交互（Interactions）]

用法

描述：使用鼠标改变元素的尺寸。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可调整尺寸（Resizable）插件让被选元素可调整尺寸（意味着它们有可拖拽的调整大小的手柄）。您可以指定一个或多个手柄，也可以指定宽度和高度的最小值也最大值。

附加说明：该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

alsoResizeanimateanimateDurationanimateEasingaspectRatioautoHidecancelco

alsoResize

类型：Selector 或 jQuery 或 Element

描述：一个或多个通过 resizable 元素进行同步调整尺寸的元素。

代码实例：

初始化带有指定 `alsoResize` 选项的 resizable：

```
$( ".selector" ).resizable({ alsoResize: "#mirror" });
```

在初始化后，获取或设置 `alsoResize` 选项：

```
// getter
var alsoResize = $( ".selector" ).resizable( "option", "alsoResize'

// setter
$( ".selector" ).resizable( "option", "alsoResize", "#mirror" );
```

默认值：false

animate

类型：Boolean

描述：调整尺寸后动态变化到最终尺寸。

代码实例：

初始化带有指定 `animate` 选项的 resizable：

```
$( ".selector" ).resizable({ animate: true });
```

在初始化后，获取或设置 `animate` 选项：

```
// getter
var animate = $( ".selector" ).resizable( "option", "animate" );

// setter
$( ".selector" ).resizable( "option", "animate", true );
```

默认值：false

animateDuration

类型：Number 或 String

描述：当使用 `[animate](#option-animate)` 选项时，动画持续的时间。

支持多个类型：

- **Number**：持续时间，以毫秒计。
- **String**：一个命名的持续时间，比如 `"slow"` 或 `"fast"` 。

代码实例：

初始化带有指定 `animateDuration` 选项的 resizable：

```
$( ".selector" ).resizable({ animateDuration: "fast" });
```

在初始化后，获取或设置 `animateDuration` 选项：

```
// getter
var animateDuration = $( ".selector" ).resizable( "option", "animat

// setter
$( ".selector" ).resizable( "option", "animateDuration", "fast" );
```

默认值："slow"

animateEasing

类型：String

描述：当使用 `[animate](#option-animate)` 选项时要使用的 Easings。

代码实例：

初始化带有指定 `animateEasing` 选项的 resizable：

```
$( ".selector" ).resizable({ animateEasing: "easeOutBounce" });
```

在初始化后，获取或设置 `animateEasing` 选项：

```
// getter
var animateEasing = $( ".selector" ).resizable( "option", "animateE

// setter
$( ".selector" ).resizable( "option", "animateEasing", "easeOutBour
```

默认值："swing"

aspectRatio

类型：Boolean 或 Number

描述：元素是否应该被限制在一个特定的长宽比。

支持多个类型：

- **Boolean**：当设置为 `true` 时，元素将保持其原有的长宽比。

- **Number**：强制在调整尺寸时元素保持特定的长宽比。

代码实例：

初始化带有指定 `aspectRatio` 选项的 resizable：

```
$( ".selector" ).resizable({ aspectRatio: true });
```

在初始化后，获取或设置 `aspectRatio` 选项：

```
// getter
var aspectRatio = $( ".selector" ).resizable( "option", "aspectRat

// setter
$( ".selector" ).resizable( "option", "aspectRatio", true );
```

默认值：false

autoHide

类型：Boolean

描述：当用户鼠标没有悬浮在元素上时是否隐藏手柄。

代码实例：

初始化带有指定 `autoHide` 选项的 resizable：

```
$( ".selector" ).resizable({ autoHide: true });
```

在初始化后，获取或设置 `autoHide` 选项：

```
// getter
var autoHide = $( ".selector" ).resizable( "option", "autoHide" );

// setter
$( ".selector" ).resizable( "option", "autoHide", true );
```

默认值：false

cancel

类型：Selector

描述：防止从指定的元素上开始调整尺寸。

代码实例：

初始化带有指定 `cancel` 选项的 resizable：

```
$( ".selector" ).resizable({ cancel: ".cancel" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).resizable( "option", "cancel" );

// setter
$( ".selector" ).resizable( "option", "cancel", ".cancel" );
```

默认值："input, textarea, button, select, option"

containment

类型：Selector 或 Element 或 String

描述：约束在指定元素或区域的边界内调整尺寸。

支持多个类型：

- **Selector**：可调整尺寸元素将被包含在 selector 第一个元素的边界内。如果未找到元素，则不设置 containment。
- **Element**：可调整尺寸元素将被包含在元素的边界内。
- **String**：可能的值： `"parent"` 、 `"document"` 。

代码实例：

初始化带有指定 `containment` 选项的 resizable：

```
$( ".selector" ).resizable({ containment: "parent" });
```

在初始化后，获取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).resizable( "option", "containmen

// setter
$( ".selector" ).resizable( "option", "containment", "parent" );
```

默认值：false

delay

类型：Number

描述：鼠标按下后直到调整尺寸开始为止的时间，以毫秒计。如果指定了该选项，调整只有在鼠标移动超过时间后才开始。该选项可以防止点击在某个元素上时不必要的调整尺寸。

代码实例：

初始化带有指定 `delay` 选项的 resizable：

```
$( ".selector" ).resizable({ delay: 150 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).resizable( "option", "delay" );

// setter
$( ".selector" ).resizable( "option", "delay", 150 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true` ，则禁用该 resizable。

代码实例：

初始化带有指定 `disabled` 选项的 resizable：

```
$( ".selector" ).resizable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).resizable( "option", "disabled" );

// setter
$( ".selector" ).resizable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后调整尺寸开始前必须移动的距离，以像素计。如果指定了该选项，调整只有在鼠标移动超过距离后才开始。该选项可以防止点击在某个元素上时不必要的调整尺寸。

代码实例：

初始化带有指定 `distance` 选项的 resizable：

```
$( ".selector" ).resizable({ distance: 30 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).resizable( "option", "distance" );

// setter
$( ".selector" ).resizable( "option", "distance", 30 );
```

默认值：1

ghost

类型：Boolean

描述：如果设置为 `true`，则为调整尺寸显示一个半透明的辅助元素。

代码实例：

初始化带有指定 `ghost` 选项的 resizable：

```
$( ".selector" ).resizable({ ghost: true });
```

在初始化后，获取或设置 `ghost` 选项：

```
// getter
var ghost = $( ".selector" ).resizable( "option", "ghost" );

// setter
$( ".selector" ).resizable( "option", "ghost", true );
```

默认值：false

grid

类型：Array

描述：把可调整尺寸元素对齐到网格，每个 x 和 y 像素。数组形式必须是 `[ x, y ]`。

代码实例：

初始化带有指定 `grid` 选项的 resizable：

```
$( ".selector" ).resizable({ grid: [ 20, 10 ] });
```

在初始化后，获取或设置 `grid` 选项：

```
// getter
var grid = $( ".selector" ).resizable( "option", "grid" );

// setter
$( ".selector" ).resizable( "option", "grid", [ 20, 10 ] );
```

默认值：false

handles

类型：String 或 Object

描述：可用于调整尺寸的处理程序。

支持多个类型：

- **String**：一个逗号分隔的列表，列表值为下面所列出的任意值：n, e, s, w, ne, se, sw, nw, all。必要的处理程序由插件自动生成。
- **Object**：支持下面的键：{ n, e, s, w, ne, se, sw, nw }。任何指定的值应该是一个匹配作为处理程序使用的 resizable 的子元素的 jQuery 选择器。

注释：当生成您自己的处理程序时，每个处理程序必须有 `ui-resizable-handle` class，也可以是适当的 `appropriate ui-resizable-{direction}` class，比如 ui-resizable-s。

代码实例：

初始化带有指定 `handles` 选项的 resizable：

```
$( ".selector" ).resizable({ handles: "n, e, s, w" });
```

在初始化后，获取或设置 `handles` 选项：

```
// getter
var handles = $( ".selector" ).resizable( "option", "handles" );

// setter
$( ".selector" ).resizable( "option", "handles", "n, e, s, w" );
```

默认值："e, s, se"

helper

类型：String

描述：一个将被添加到代理元素的 class 名称，用于描绘调整手柄拖拽过程中调整的轮廓。一旦调整完成，原来的元素则被重新定义尺寸。

代码实例：

初始化带有指定 `helper` 选项的 resizable：

```
$( ".selector" ).resizable({ helper: "resizable-helper" });
```

在初始化后，获取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).resizable( "option", "helper" );

// setter
$( ".selector" ).resizable( "option", "helper", "resizable-helper"
```

默认值：false

maxHeight

类型：Number

描述：resizable 允许被调整到的最大高度。

代码实例：

初始化带有指定 `maxHeight` 选项的 resizable：

```
$( ".selector" ).resizable({ maxHeight: 300 });
```

在初始化后，获取或设置 `maxHeight` 选项：

```
// getter
var maxHeight = $( ".selector" ).resizable( "option", "maxHeight" 

// setter
$( ".selector" ).resizable( "option", "maxHeight", 300 );
```

默认值：null

maxWidth

类型：Number

描述：resizable 允许被调整到的最大宽度。

代码实例：

初始化带有指定 `maxWidth` 选项的 resizable：

```
$( ".selector" ).resizable({ maxWidth: 300 });
```

在初始化后，获取或设置 `maxWidth` 选项：

```
// getter
var maxWidth = $( ".selector" ).resizable( "option", "maxWidth" );

// setter
$( ".selector" ).resizable( "option", "maxWidth", 300 );
```

默认值：null

minHeight

类型：Number

描述：resizable 允许被调整到的最小高度。

代码实例：

初始化带有指定 `minHeight` 选项的 resizable：

```
$( ".selector" ).resizable({ minHeight: 150 });
```

在初始化后，获取或设置 `minHeight` 选项：

```
// getter
var minHeight = $( ".selector" ).resizable( "option", "minHeight" );

// setter
$( ".selector" ).resizable( "option", "minHeight", 150 );
```

默认值：10

minWidth

类型：Number

描述：resizable 允许被调整到的最小宽度。

代码实例：

初始化带有指定 `minWidth` 选项的 resizable：

```
$( ".selector" ).resizable({ minWidth: 150 });
```

在初始化后，获取或设置 `minWidth` 选项：

```
// getter
var minWidth = $( ".selector" ).resizable( "option", "minWidth" );

// setter
$( ".selector" ).resizable( "option", "minWidth", 150 );
```

默认值：10

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 resizable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).resizable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 resizable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).resizable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 resizable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).resizable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).resizable( "option", "disabled" );
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 resizable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).resizable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 resizable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).resizable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 resizable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).resizable( "option", { disabled: true } );
```

widget()

类型：jQuery

描述：返回一个包含 resizable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).resizable( "widget" );
```

事件

create(event, ui)

类型：resizecreate

描述：当 resizable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 resizable：

```
$( ".selector" ).resizable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 resizecreate 事件：

```
$( ".selector" ).on( "resizecreate", function( event, ui ) {} );
```

resize(event, ui)

类型：resize

描述：在调整尺寸期间当调整手柄拖拽时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **element** 类型：jQuery 描述：jQuery 对象，表示要被调整尺寸的元素。
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被调整尺寸的助手（helper）。
    - **originalElement** 类型：jQuery 描述：jQuery 对象，表示被包裹之前的原始元素。
    - **originalPosition** 类型：Object 描述：resizable 调整前的位置，表示为 `{ top, left }`。
    - **originalSize** 类型：Object 描述：resizable 调整前的尺寸，表示为 `{ width, height }`。
    - **position** 类型：Object 描述：当前位置，表示为 `{ top, left }`。
    - **size** 类型：Object 描述：当前位置，表示为 `{ width, height }`。

代码实例：

初始化带有指定 resize 回调的 resizable：

```
$( ".selector" ).resizable({
  resize: function( event, ui ) {}
});
```

绑定一个事件监听器到 resize 事件：

```
$( ".selector" ).on( "resize", function( event, ui ) {} );
```

start(event, ui)

类型：resizestart

描述：当调整尺寸操作开始时触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **element** 类型：jQuery 描述：jQuery 对象，表示要被调整尺寸的元素。
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被调整尺寸的助手（helper）。
    - **originalElement** 类型：jQuery 描述：jQuery 对象，表示被包裹之前的原始元素。
    - **originalPosition** 类型：Object 描述：resizable 调整前的位置，表示为 `{ top, left }`。
    - **originalSize** 类型：Object 描述：resizable 调整前的尺寸，表示为 `{ width, height }`。

- **position** 类型：Object 描述：当前位置，表示为 `{ top, left }` 。
- **size** 类型：Object 描述：当前位置，表示为 `{ width, height }` 。

代码实例：

初始化带有指定 start 回调的 resizable：

```
$( ".selector" ).resizable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 resizestart 事件：

```
$( ".selector" ).on( "resizestart", function( event, ui ) {} );
```

stop(event, ui)

类型：resizestop

描述：当调整尺寸操作停止时触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **element** 类型：jQuery 描述：jQuery 对象，表示要被调整尺寸的元素。
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被调整尺寸的助手（helper）。
  - **originalElement** 类型：jQuery 描述：jQuery 对象，表示被包裹之前的原始元素。
  - **originalPosition** 类型：Object 描述：resizable 调整前的位置，表示为 `{ top, left }` 。
  - **originalSize** 类型：Object 描述：resizable 调整前的尺寸，表示为 `{ width, height }` 。
  - **position** 类型：Object 描述：当前位置，表示为 `{ top, left }` 。
  - **size** 类型：Object 描述：当前位置，表示为 `{ width, height }` 。

代码实例：

初始化带有指定 stop 回调的 resizable：

```
$( ".selector" ).resizable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 resizestop 事件：

```
$( ".selector" ).on( "resizestop", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 可调整尺寸小部件（Resizable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可调整尺寸小部件（Resizable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #resizable {
    width: 100px;
    height: 100px;
    background: #ccc;
}    </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="resizable"></div>

<script>
$( "#resizable" ).resizable();
</script>

</body>
</html>
```

jQuery UI API - 可选择小部件（Selectable Widget）

所属类别

交互（Interactions）

用法

描述：使用鼠标选择单个元素或一组元素。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可选择（Selectable）插件允许通过鼠标拖拽选择元素（有时被称为一个套索）。可以在按住 ctrl/meta 键的同时单击或拖动来选择多个（不连续的）元素。

附加说明：该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

选项	
appendToautoRefreshcanceldelaydisableddistancefiltertolerance	destroydisab

appendTo

类型：Selector

描述：选择助手（套索）要被添加到哪一个元素。

代码实例：

初始化带有指定 `appendTo` 选项的 draggable：

```
$( ".selector" ).selectable({ appendTo: "#someElem" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).selectable( "option", "appendTo" );

// setter
$( ".selector" ).selectable( "option", "appendTo", "#someElem" );
```

默认值："body"

autoRefresh

类型：Boolean

描述：该选项决定是否在每个选择操作的开始时更新（重新计算）每个选择项的位置和尺寸。如果您有多个项目，您可能要设置该选项为 false，并手动调用 `[refresh()](#method-refresh)` 方法。

代码实例：

初始化带有指定 `autoRefresh` 选项的 draggable：

```
$( ".selector" ).selectable({ autoRefresh: false });
```

在初始化后，获取或设置 `autoRefresh` 选项：

```
// getter
var autoRefresh = $( ".selector" ).selectable( "option", "autoRefre

// setter
$( ".selector" ).selectable( "option", "autoRefresh", false );
```

默认值：true

cancel

类型：Selector

描述：防止从匹配选择器的元素上开始选择。

代码实例：

初始化带有指定 `cancel` 选项的 selectable：

```
$( ".selector" ).selectable({ cancel: "a,.cancel" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).selectable( "option", "cancel" );

// setter
$( ".selector" ).selectable( "option", "cancel", "a,.cancel" );
```

默认值："input, textarea, button, select, option"

delay

类型：Number

描述：鼠标按下后直到选择开始的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的选择。

代码实例：

初始化带有指定 `delay` 选项的 selectable：

```
$( ".selector" ).selectable({ delay: 150 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).selectable( "option", "delay" );

// setter
$( ".selector" ).selectable( "option", "delay", 150 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 selectable。

代码实例：

初始化带有指定 `disabled` 选项的 selectable：

```
$( ".selector" ).selectable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).selectable( "option", "disabled" );

// setter
$( ".selector" ).selectable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后选择开始前必须移动的距离，以像素计。如果指定了该选项，选择只有在鼠标拖拽超出指定距离时才会开始。该选项可以防止点击在某个元素上时不必要的选择。

代码实例：

初始化带有指定 `distance` 选项的 selectable：

```
$( ".selector" ).selectable({ distance: 30 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).selectable( "option", "distance" );

// setter
$( ".selector" ).selectable( "option", "distance", 30 );
```

默认值：0

filter

类型：Selector

描述：要制作选择项（可被选择的）的匹配的子元素。

代码实例：

初始化带有指定 `filter` 选项的 selectable：

```
$( ".selector" ).selectable({ filter: "li" });
```

在初始化后，获取或设置 `filter` 选项：

```
// getter
var filter = $( ".selector" ).selectable( "option", "filter" );

// setter
$( ".selector" ).selectable( "option", "filter", "li" );
```

默认值："*"

tolerance

类型：String

描述：指定用于测试套索是否选择一个项目的模式。可能的值：

- `"fit"` ：套索完全重叠在项目上。
- `"touch"` ：套索重叠在项目上，任何比例皆可。

代码实例：

初始化带有指定 `tolerance` 选项的 selectable：

```
$( ".selector" ).selectable({ tolerance: "fit" });
```

在初始化后，获取或设置 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).selectable( "option", "tolerance"

// setter
$( ".selector" ).selectable( "option", "tolerance", "fit" );
```

默认值："touch"

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 selectable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).selectable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 selectable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).selectable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 selectable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).selectable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).selectable( "option", "disabled"
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 selectable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).selectable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 selectable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).selectable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 selectable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).selectable( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：更新每个选择项元素的位置和尺寸。当 `[autoRefresh](#option-autoRefresh)` 选项被设置为 false 时，该方法可用于手动重新计算每个选择项的位置和尺寸。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).selectable( "refresh" );
```

widget()

类型：jQuery

描述：返回一个包含 selectable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).selectable( "widget" );
```

事件

create(event, ui)

类型：selectablecreate

描述：当 selectable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 selectable：

```
$( ".selector" ).selectable({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectablecreate 事件：

```
$( ".selector" ).on( "selectablecreate", function( event, ui ) {}
```

selected(event, ui)

类型：selectableselected

描述：当每个元素被添加选择时，在选择操作结尾触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **selected** 类型：Element 描述：被选择的可选择项目。

代码实例：

初始化带有指定 selected 回调的 selectable：

```
$( ".selector" ).selectable({
  selected: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableselected 事件：

```
$( ".selector" ).on( "selectableselected", function( event, ui ) {
```

selecting(event, ui)

类型：selectableselecting

描述：当每个元素被添加选择时，在选择操作期间触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **selecting** 类型：Element 描述：正被选择的当前可选择项目。

代码实例：

初始化带有指定 selecting 回调的 selectable：

```
$( ".selector" ).selectable({
  selecting: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableselecting 事件：

```
$( ".selector" ).on( "selectableselecting", function( event, ui ) 
```

start(event, ui)

类型：selectablestart

描述：在选择操作开头触发。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 start 回调的 selectable：

```
$( ".selector" ).selectable({
  start: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectablestart 事件：

```
$( ".selector" ).on( "selectablestart", function( event, ui ) {} )
```

stop(event, ui)

类型：selectablestop

描述：在选择操作结尾触发。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 stop 回调的 selectable：

```
$( ".selector" ).selectable({
  stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectablestop 事件：

```
$( ".selector" ).on( "selectablestop", function( event, ui ) {} );
```

unselected(event, ui)

类型：selectableunselected

描述：当每个元素从选择中被移除时，在选择操作结尾触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **unselected** 类型：Element 描述：未被选择的可选择项目。

代码实例：

初始化带有指定 unselected 回调的 selectable：

```
$( ".selector" ).selectable({
  unselected: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableunselected 事件：

```
$( ".selector" ).on( "selectableunselected", function( event, ui )
```

unselecting(event, ui)

类型：selectableunselecting

描述：当每个元素从选择中被移除时，在选择操作期间触发。

- **event** 类型：Event
- **ui** 类型：Object
    - **unselecting** 类型：Element 描述：正未被选择的当前可选择项目。

代码实例：

初始化带有指定 unselecting 回调的 selectable：

```
$( ".selector" ).selectable({
  unselecting: function( event, ui ) {}
});
```

绑定一个事件监听器到 selectableunselecting 事件：

```
$( ".selector" ).on( "selectableunselecting", function( event, ui ) 
```

实例

一个简单的 jQuery UI 可选择小部件（Selectable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可选择小部件（Selectable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  #selectable .ui-selecting {
    background: #ccc;
  }
  #selectable .ui-selected {
    background: #999;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="selectable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>

<script>
$( "#selectable" ).selectable();
</script>

</body>
</html>
```

jQuery UI API - 可排序小部件（Sortable Widget）

所属类别

交互（Interactions）

用法

描述：使用鼠标调整列表中或者网格中元素的排序。

版本新增：1.0

依赖：

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

注释：jQuery UI 可排序（Sortable）插件让被选元素通过鼠标拖拽进行排序。

注释：为了排序表格行， `tbody` 必须是可排序的（sortable）， 而不是 `table` 。

快速导航

appendToaxiscancelconnectWithcontainmentcursorcursorAtdelaydisableddistanc

appendTo

类型：jQuery 或 Element 或 Selector 或 String

描述：当拖拽时，通过鼠标移动的助手被追加到哪里（例如，解决 overlap/zIndex 问题）。

支持多个类型：

- **jQuery**：一个 jQuery 对象，包含助手（helper）要追加到的元素。
- **Element**：要追加助手（helper）的元素。
- **Selector**：一个选择器，指定哪个元素要追加助手（helper）。
- **String**：字符串 `"parent"` 将促使助手（helper）成为 sortable 项目的同级。

代码实例：

初始化带有指定 `appendTo` 选项的 sortable：

```
$( ".selector" ).sortable({ appendTo: document.body });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).sortable( "option", "appendTo" );

// setter
$( ".selector" ).sortable( "option", "appendTo", document.body );
```

默认值："parent"

axis

类型：String

描述：如果定义了该选项，项目只能在水平或垂直方向上被拖拽。可能的值：`"x"`，`"y"`。

代码实例：

初始化带有指定 `axis` 选项的 sortable：

```
$( ".selector" ).sortable({ axis: "x" });
```

在初始化后，获取或设置 `axis` 选项：

```
// getter
var axis = $( ".selector" ).sortable( "option", "axis" );

// setter
$( ".selector" ).sortable( "option", "axis", "x" );
```

默认值：false

cancel

类型：Selector

描述：防止从匹配选择器的元素上开始排序。

代码实例：

初始化带有指定 `cancel` 选项的 sortable：

```
$( ".selector" ).sortable({ cancel: "a,button" });
```

在初始化后，获取或设置 `cancel` 选项：

```
// getter
var cancel = $( ".selector" ).sortable( "option", "cancel" );

// setter
$( ".selector" ).sortable( "option", "cancel", "a,button" );
```

默认值："input, textarea, button, select, option"

connectWith

类型：Selector

描述：列表中的项目需被连接的其他 sortable 元素的选择器。这是一个单向关系，如果您想要项目被双向连接，必须在两个 sortable 元素上都设置 `connectWith` 选项。

代码实例：

初始化带有指定 `connectWith` 选项的 sortable：

```
$( ".selector" ).sortable({ connectWith: "#shopping-cart" });
```

在初始化后，获取或设置 `connectWith` 选项：

```
// getter
var connectWith = $( ".selector" ).sortable( "option", "connectWit

// setter
$( ".selector" ).sortable( "option", "connectWith", "#shopping-car
```

默认值：false

containment

类型：Element 或 Selector 或 String

描述：定义拖拽时，sortable 项目被约束的边界。

注释：为 containment 指定的元素必须有一个已计算的宽度和高度（尽管它不需要显式）。例如，如果您有 `float: left` sortable 子元素，并指定了 `containment: "parent"` ，请确保在 sortable/parent 容器上有 `float: left` ，否则它将有 `height: 0` ，导致未定义的行为。

支持多个类型：

- **Element**：一个要作为容器使用的元素。
- **Selector**：一个选择器，指定一个要作为容器使用的元素。
- **String**：一个字符串，标识一个要作为容器使用的元素。可能的值： `"parent"` 、 `"document"` 、 `"window"` 。

代码实例：

初始化带有指定 `containment` 选项的 sortable：

```
$( ".selector" ).sortable({ containment: "parent" });
```

在初始化后，获取或设置 `containment` 选项：

```
// getter
var containment = $( ".selector" ).sortable( "option", "containment

// setter
$( ".selector" ).sortable( "option", "containment", "parent" );
```

默认值：false

cursor

类型：String

描述：定义当排序时被显示的光标。

代码实例：

初始化带有指定 `cursor` 选项的 sortable：

```
$( ".selector" ).sortable({ cursor: "move" });
```

在初始化后，获取或设置 `cursor` 选项：

```
// getter
var cursor = $( ".selector" ).sortable( "option", "cursor" );

// setter
$( ".selector" ).sortable( "option", "cursor", "move" );
```

默认值："auto"

cursorAt

类型：Object

描述：移动排序元素或助手（helper），这样光标总是出现，以便从相同的位置进行拖拽。坐标可通过一个或两个键的组合成一个哈希给出：`{ top, left, right, bottom }`。

代码实例：

初始化带有指定 `cursorAt` 选项的 sortable：

```
$( ".selector" ).sortable({ cursorAt: { left: 5 } });
```

在初始化后，获取或设置 `cursorAt` 选项：

```
// getter
var cursorAt = $( ".selector" ).sortable( "option", "cursorAt" );

// setter
$( ".selector" ).sortable( "option", "cursorAt", { left: 5 } );
```

默认值：false

delay

类型：Integer

描述：鼠标按下后直到排序开始的时间，以毫秒计。该选项可以防止点击在某个元素上时不必要的拖拽。

代码实例：

初始化带有指定 `delay` 选项的 sortable：

```
$( ".selector" ).sortable({ delay: 150 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).sortable( "option", "delay" );

// setter
$( ".selector" ).sortable( "option", "delay", 150 );
```

默认值：0

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 sortable。

代码实例：

初始化带有指定 `disabled` 选项的 sortable：

```
$( ".selector" ).sortable({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).sortable( "option", "disabled" );

// setter
$( ".selector" ).sortable( "option", "disabled", true );
```

默认值：false

distance

类型：Number

描述：鼠标按下后排序开始前必须移动的距离，以像素计。如果指定了该选项，排序只有在鼠标拖拽超出指定距离时才会开始。该选项可以用于允许在一个手柄内的元素上点击。

代码实例：

初始化带有指定 `distance` 选项的 sortable：

```
$( ".selector" ).sortable({ distance: 5 });
```

在初始化后，获取或设置 `distance` 选项：

```
// getter
var distance = $( ".selector" ).sortable( "option", "distance" );

// setter
$( ".selector" ).sortable( "option", "distance", 5 );
```

默认值：1

dropOnEmpty

类型：Boolean

描述：如果为 `false`，从该 sortable 的项目不能被放置在空连接的 sortable 上（请查看 `[connectWith](#option-connectWith)` 选项）。

代码实例：

初始化带有指定 `dropOnEmpty` 选项的 sortable：

```
$( ".selector" ).sortable({ dropOnEmpty: false });
```

在初始化后，获取或设置 `dropOnEmpty` 选项：

```
// getter
var dropOnEmpty = $( ".selector" ).sortable( "option", "dropOnEmpty

// setter
$( ".selector" ).sortable( "option", "dropOnEmpty", false );
```

默认值：true

forceHelperSize

类型：Boolean

描述：如果为 `true`，强制助手（helper）有一个尺寸。

代码实例：

初始化带有指定 `forceHelperSize` 选项的 sortable：

```
$( ".selector" ).sortable({ forceHelperSize: true });
```

在初始化后，获取或设置 `forceHelperSize` 选项：

```
// getter
var forceHelperSize = $( ".selector" ).sortable( "option", "forceHe

// setter
$( ".selector" ).sortable( "option", "forceHelperSize", true );
```

默认值：false

forcePlaceholderSize

类型：Boolean

描述：如果为 `true`，强制占位符（placeholder）有一个尺寸。

代码实例：

初始化带有指定 `forcePlaceholderSize` 选项的 sortable：

```
$( ".selector" ).sortable({ forcePlaceholderSize: true });
```

在初始化后，获取或设置 `forcePlaceholderSize` 选项：

```
// getter
var forcePlaceholderSize = $( ".selector" ).sortable( "option", "fo

// setter
$( ".selector" ).sortable( "option", "forcePlaceholderSize", true )
```

默认值：false

grid

类型：Array

描述：对齐排序元素或助手（helper）到网格，每个 x 和 y 像素。数组形式必须是 `[ x, y ]`。

代码实例：

初始化带有指定 `grid` 选项的 sortable：

```
$( ".selector" ).sortable({ grid: [ 20, 10 ] });
```

在初始化后，获取或设置 `grid` 选项：

```
// getter
var grid = $( ".selector" ).sortable( "option", "grid" );

// setter
$( ".selector" ).sortable( "option", "grid", [ 20, 10 ] );
```

默认值：false

handle

类型：Selector 或 Element

描述：如果指定了该选项，则限制在指定的元素上开始排序。

代码实例：

初始化带有指定 `handle` 选项的 sortable：

```
$( ".selector" ).sortable({ handle: ".handle" });
```

在初始化后，获取或设置 `handle` 选项：

```
// getter
var handle = $( ".selector" ).sortable( "option", "handle" );

// setter
$( ".selector" ).sortable( "option", "handle", ".handle" );
```

默认值：false

helper

类型：String 或 Function()

描述：允许一个 helper 元素用于拖拽显示。

支持多个类型：

- **String**：如果设置为 `"clone"`，元素将被克隆，且克隆将被拖拽。
- **Function**：一个函数，将返回拖拽时要使用的 DOMElement。函数接收事件，且元素正被排序。

代码实例：

初始化带有指定 `helper` 选项的 sortable：

```
$( ".selector" ).sortable({ helper: "clone" });
```

在初始化后，获取或设置 `helper` 选项：

```
// getter
var helper = $( ".selector" ).sortable( "option", "helper" );

// setter
$( ".selector" ).sortable( "option", "helper", "clone" );
```

默认值："original"

items

类型：Selector

描述：指定元素内的哪一个项目应是 sortable。

代码实例：

初始化带有指定 `items` 选项的 sortable：

```
$( ".selector" ).sortable({ items: "&gt; li" });
```

在初始化后，获取或设置 `items` 选项：

```
// getter
var items = $( ".selector" ).sortable( "option", "items" );

// setter
$( ".selector" ).sortable( "option", "items", "&gt; li" );
```

默认值："> *"

opacity

类型：Number

描述：当排序时助手（helper）的不透明度。从 `0.01` 到 `1` 。

代码实例：

初始化带有指定 `opacity` 选项的 sortable：

```
$( ".selector" ).sortable({ opacity: 0.5 });
```

在初始化后，获取或设置 `opacity` 选项：

```
// getter
var opacity = $( ".selector" ).sortable( "option", "opacity" );

// setter
$( ".selector" ).sortable( "option", "opacity", 0.5 );
```

默认值：false

placeholder

类型：String

描述：要应用的 class 名称，否则为白色空白。

代码实例：

初始化带有指定 `placeholder` 选项的 sortable：

```
$( ".selector" ).sortable({ placeholder: "sortable-placeholder" });
```

在初始化后，获取或设置 `placeholder` 选项：

```
// getter
var placeholder = $( ".selector" ).sortable( "option", "placeholder

// setter
$( ".selector" ).sortable( "option", "placeholder", "sortable-place
```

默认值：false

revert

类型：Boolean 或 Number

描述：sortable 项目是否使用一个流畅的动画还原到它的新位置。

支持多个类型：

- **Boolean**：当设置为 `true`，该项目将会使用动画，动画使用默认的持续时间。
- **Number**：动画的持续时间，以毫秒计。

代码实例：

初始化带有指定 `revert` 选项的 sortable：

```
$( ".selector" ).sortable({ revert: true });
```

在初始化后，获取或设置 `revert` 选项：

```
// getter
var revert = $( ".selector" ).sortable( "option", "revert" );

// setter
$( ".selector" ).sortable( "option", "revert", true );
```

默认值：false

scroll

类型：Boolean

描述：如果设置为 `true`，当到达边缘时页面会滚动。

代码实例：

初始化带有指定 `scroll` 选项的 sortable：

```
$( ".selector" ).sortable({ scroll: false });
```

在初始化后，获取或设置 `scroll` 选项：

```
// getter
var scroll = $( ".selector" ).sortable( "option", "scroll" );

// setter
$( ".selector" ).sortable( "option", "scroll", false );
```

默认值：true

scrollSensitivity

类型：Number

描述：定义鼠标距离边缘多少距离时开始滚动。

代码实例：

初始化带有指定 `scrollSensitivity` 选项的 sortable：

```
$( ".selector" ).sortable({ scrollSensitivity: 10 });
```

在初始化后，获取或设置 `scrollSensitivity` 选项：

```
// getter
var scrollSensitivity = $( ".selector" ).sortable( "option", "scrol

// setter
$( ".selector" ).sortable( "option", "scrollSensitivity", 10 );
```

默认值：20

scrollSpeed

类型：Number

描述：当鼠标指针获取到在 `[scrollSensitivity](#option-scrollSensitivity)` 距离内时，窗体滚动的速度。如果 `scroll` 选项是 `false` 则忽略。

代码实例：

初始化带有指定 `scrollSpeed` 选项的 sortable：

```
$( ".selector" ).sortable({ scrollSpeed: 40 });
```

在初始化后，获取或设置 `scrollSpeed` 选项：

```
// getter
var scrollSpeed = $( ".selector" ).sortable( "option", "scrollSpeed

// setter
$( ".selector" ).sortable( "option", "scrollSpeed", 40 );
```

默认值：20

tolerance

类型：String

描述：指定用于测试项目被移动时是否覆盖在另一个项目上的模式。可能的值：

- `"intersect"` ：项目至少 50% 重叠在其他项目上。
- `"pointer"` ：鼠标指针重叠在其他项目上。

代码实例：

初始化带有指定 `tolerance` 选项的 sortable：

```
$( ".selector" ).sortable({ tolerance: "pointer" });
```

在初始化后，获取或设置 `tolerance` 选项：

```
// getter
var tolerance = $( ".selector" ).sortable( "option", "tolerance" );

// setter
$( ".selector" ).sortable( "option", "tolerance", "pointer" );
```

默认值："intersect"

zIndex

类型：Integer

描述：当被排序时，元素/助手（helper）的 Z-index。

代码实例：

初始化带有指定 `zIndex` 选项的 sortable：

```
$( ".selector" ).sortable({ zIndex: 9999 });
```

在初始化后，获取或设置 `zIndex` 选项：

```
// getter
var zIndex = $( ".selector" ).sortable( "option", "zIndex" );

// setter
$( ".selector" ).sortable( "option", "zIndex", 9999 );
```

默认值：1000

方法

cancel()

类型：jQuery (plugin only)

描述：当前排序开始时，取消一个在当前 sortable 中的改变，且恢复到之前的状态。在 stop 和 receive 回调函数中非常有用。

- 该方法不接受任何参数。

代码实例：

调用 cancel 方法：

```
$( ".selector" ).sortable( "cancel" );
```

destroy()

类型：jQuery (plugin only)

描述：完全移除 sortable 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).sortable( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 sortable。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).sortable( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 sortable。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).sortable( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).sortable( "option", "disabled" );
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 draggable 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).sortable( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 sortable 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).sortable( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 sortable 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).sortable( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：刷新 sortable 项目。触发所有 sortable 项目重新加载，导致新的项目被认可。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).sortable( "refresh" );
```

refreshPositions()

类型：jQuery (plugin only)

描述：刷新 sortable 项目的缓存位置。调用该方法刷新所有 sortable 的已缓存的项目位置。

- 该方法不接受任何参数。

代码实例：

调用 refreshPositions 方法：

```
$( ".selector" ).sortable( "refreshPositions" );
```

serialize(options)

类型：String

描述：序列化 sortable 的项目 `id` 为一个 form/ajax 可提交的字符串。调用该方法会产生一个可被追加到任何 url 中的哈希，以便简单地把一个新的项目顺序提交回服务器。

默认情况下，它通过每个项目的 `id` 进行工作，id 格式为 `"setname_number"`，且它会产生一个形如 `"setname[]=number&setname[]=number"` 的哈希。

注释：如果序列化返回一个空的字符串，请确认 `id` 属性包含一个下划线（）。形式必须是 *`"set_number"`* 。例如，一个 *`id`* 属性为 *`"foo_1"`* 、 *`"foo_5"`* 、 *`"foo_2"`* 的 *3* 元素列表将序列化为 *`"foo[]=1&foo[]=5&foo[]=2"`* 。您可以使用下划线（）、等号（=）或连字符（-）来分隔集合和数字。例如， `"foo=1"` 、 `"foo-1"` 、 `"foo_1"` 所有都序列化为 `"foo[]=1"` 。

- **options** 类型：Object 要自定义序列化的选项。
  - **key**（默认值： `属性中分隔符前面的部分` ） 类型：String 描述：把 `part1[]` 替换为指定的值。
  - **attribute**（默认值： `"id"` ） 类型：String 描述：值要使用的属性名称。
  - **expression**（默认值： `/(.+)[-=_](.+)/` ） 类型：RegExp 描述：用于把属性值分隔为键和值两部分的正则表达式。

代码实例：

调用 serialize 方法：

```
var sorted = $( ".selector" ).sortable( "serialize", { key: "sort"
```

toArray(options)

类型：Array

描述：序列化 sortable 的项目 `id` 为一个字符串的数组。

- **options** 类型：Object 要自定义序列化的选项。
    - **attribute**（默认值：`"id"`） 类型：String 描述：值要使用的属性名称。

代码实例：

调用 toArray 方法：

```
var sortedIDs = $( ".selector" ).sortable( "toArray" );
```

widget()

类型：jQuery

描述：返回一个包含 sortable 元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).sortable( "widget" );
```

事件

activate(event, ui)

类型：sortactivate

描述：当使用被连接列表时触发该事件，每个被连接列表在拖拽开始时接收它。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为

  `{ top, left }`。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 activate 回调的 sortable：

```
$( ".selector" ).sortable({
activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortactivate 事件：

```
$( ".selector" ).on( "sortactivate", function( event, ui ) {} );
```

beforeStop(event, ui)

类型：sortbeforestop

描述：当排序停止时触发该事件，除了当占位符（placeholder）/助手（helper）仍然可用时。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }`。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 beforeStop 回调的 sortable：

```
$( ".selector" ).sortable({
beforeStop: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortbeforestop 事件：

```
$( ".selector" ).on( "sortbeforestop", function( event, ui ) {} );
```

change(event, ui)

类型：sortchange

描述：在排序期间触发该事件，除了当 DOM 位置改变时。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 change 回调的 sortable：

```
$( ".selector" ).sortable({
change: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortchange 事件：

```
$( ".selector" ).on( "sortchange", function( event, ui ) {} );
```

create(event, ui)

类型：sortcreate

描述：当 droppable 被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 sortable：

```
$( ".selector" ).sortable({
create: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortcreate 事件：

```
$( ".selector" ).on( "sortcreate", function( event, ui ) {} );
```

deactivate(event, ui)

类型：sortdeactivate

描述：当排序停止时触发该事件，该事件传播到所有可能的连接列表。。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 deactivate 回调的 sortable：

```
$( ".selector" ).sortable({
deactivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortdeactivate 事件：

```
$( ".selector" ).on( "sortdeactivate", function( event, ui ) {} );
```

out(event, ui)

类型：sortout

描述：当一个 sortable 项目从一个 sortable 列表移除时触发该事件。

注释：当一个 sortable 项目被撤销时也会触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 out 回调的 sortable：

```
$( ".selector" ).sortable({
out: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortout 事件：

```
$( ".selector" ).on( "sortout", function( event, ui ) {} );
```

over(event, ui)

类型：sortover

描述：当一个 sortable 项目移动到一个 sortable 列表时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }`。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }`。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }`。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 over 回调的 sortable：

```
$( ".selector" ).sortable({
out: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortover 事件：

```
$( ".selector" ).on( "sortout", function( event, ui ) {} );
```

receive(event, ui)

类型：sortreceive

描述：当来自一个连接的 sortable 列表的一个项目被放置到另一个列表时触发该事件。后者是事件目标。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为

    - `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 receive 回调的 sortable：

```
$( ".selector" ).sortable({
receive: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortreceive 事件：

```
$( ".selector" ).on( "sortreceive", function( event, ui ) {} );
```

remove(event, ui)

类型：sortremove

描述：当来自一个连接的 sortable 列表的一个项目被放置到另一个列表时触发该事件。前者是事件目标。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 remove 回调的 sortable：

```
$( ".selector" ).sortable({
remove: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortremove 事件：

```
$( ".selector" ).on( "sortremove", function( event, ui ) {} );
```

sort(event, ui)

类型：sort

描述：在排序期间触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
    - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
    - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
    - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
    - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
    - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
    - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
    - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 sort 回调的 sortable：

```
$( ".selector" ).sortable({
sort: function( event, ui ) {}
});
```

绑定一个事件监听器到 sort 事件：

```
$( ".selector" ).on( "sort", function( event, ui ) {} );
```

start(event, ui)

类型：sortstart

描述：当排序开始时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
  - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
  - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
  - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 start 回调的 sortable：

```
$( ".selector" ).sortable({
start: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortstart 事件：

```
$( ".selector" ).on( "sortstart", function( event, ui ) {} );
```

stop(event, ui)

类型：sortstop

描述：当排序停止时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
  - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
  - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
  - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。

   - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
   - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
   - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 stop 回调的 sortable：

```
$( ".selector" ).sortable({
stop: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortstop 事件：

```
$( ".selector" ).on( "sortstop", function( event, ui ) {} );
```

update(event, ui)

类型：sortupdate

描述：当用户停止排序且 DOM 位置改变时触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
   - **helper** 类型：jQuery 描述：jQuery 对象，表示被排序的助手（helper）。
   - **item** 类型：jQuery 描述：jQuery 对象，表示当前被拖拽的元素。
   - **offset** 类型：Object 描述：助手（helper）的当前的绝对位置，表示为 `{ top, left }` 。
   - **position** 类型：Object 描述：助手（helper）的当前位置，表示为 `{ top, left }` 。
   - **originalPosition** 类型：Object 描述：元素的原始位置，表示为 `{ top, left }` 。
   - **sender** 类型：jQuery 描述：如果从一个 sortable 移动到另一个 sortable，项目来自的那个 sortable。
   - **placeholder** 类型：jQuery 描述：jQuery 对象，表示被作为占位符使用的元素。

代码实例：

初始化带有指定 update 回调的 sortable：

```
$( ".selector" ).sortable({
update: function( event, ui ) {}
});
```

绑定一个事件监听器到 sortupdate 事件：

```
$( ".selector" ).on( "sortupdate", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 可排序小部件（Sortable Widget）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>可排序小部件（Sortable Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="sortable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>

<script>$("#sortable").sortable();</script>

</body>
</html>
```

jQuery UI API 类别 - 方法重载（Method Overrides）

jQuery UI 重载了几个内置的 jQuery 方法，以提供额外的功能。当使用这些重载时，确保加载 jQuery UI 是很重要的。如果 jQuery UI 未加载，方法依然存在，但预期的功能将不可用，这会导致难以追踪的错误。

API	描述	也属于类别
.addClass()	当动画样式改变时，为匹配的元素集合内的每个元素添加指定的 Class。	特效（Effects） | 特效核心（Effects Core）
.focus()	异步聚焦到一个元素。	方法（Method） | UI 核心（UI Core）
.hide()	使用自定义效果来隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法（Method）
.position()	相对另一个元素定位一个元素。	方法（Method） | 实用工具（Utilities）
.removeClass()	当动画样式改变时，为匹配的元素集合内的每个元素移除指定的 Class。	特效（Effects） | 特效核心（Effects Core）
.show()	使用自定义效果来显示匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法（Method）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法（Method）
.toggleClass()	当动画样式改变时，根据 Class 是否存在以及 switch 参数的值，为匹配的元素集合内的每个元素添加或移除一个或多个 Class。	特效（Effects） | 特效核心（Effects Core）

jQuery UI API - .focus()

所属类别

方法重载（Method Overrides） | 方法（Methods） | UI 核心（UI Core）

用法

描述：异步聚焦到一个元素。

返回：jQuery

```
.focus( delay [, callback ] )Returns: jQuery
```

参数	类型	描述
delay	Integer	聚焦前等待的毫秒数。
callback	Function()	元素被聚焦后要调用的函数。

该插件扩展自 jQuery 内置的 .focus() 方法。如果 jQuery UI 未加载，调用 `.focus()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

jQuery UI API - .position()

所属类别

方法重载（Method Overrides） | 方法（Methods） | 实用工具（Utilities）

用法

描述：相对另一个元素定位一个元素。

返回：jQuery

版本新增：1.8

```
.position( options )
```

参数	类型	描述
options	Object	

- **my**（默认值：`"center"`） 类型：String 描述：定义被定位元素上对准目标元素的位置："horizontal vertical" 对齐方式。一个单一的值，比如 "right" 将规范为 "right center"，"top" 将规范为 "center top"（下面的 CSS 公约）。可接受的水平值："left"、"center"、"right"。可接受的垂直值："top"、"center"、"bottom"。例如，"left top" 或 "center center"。每个纬度也可以包含偏移，以像素计或以百分比计，例如 "right+10 top-25%"。百分比偏移是相对于被定位的元素。
- **at**（默认值：`"center"`） 类型：String 描述：定义目标元素上对准被定位元素的位置： "horizontal vertical" 对齐方式。如需了解更多细节请查看 my 选项上的可能值。百分比偏移是相对于目标元素。
- **of**（默认值：`null`） 类型：Selector 或 Element 或 jQuery 或 Event 描述：要定位的元素。如果您提供一个选择器（Selector）或 jQuery 对象，将使用第一个匹配元素。如果您提供一个事件（Event）对象，将使用 pageX 和 pageY 属性，例如 "#top-menu"。
- **collision**（默认值：`"flip"`） 类型：String 描述：当被定位元素在某些方向上溢出窗口，则移动它到另一个位置。与 my 和 at 选项相似，该选项会接受一个单一的值或一对 horizontal/vertical 值。例如："flip"、"fit"、"fit flip"、"fit none"。
    - "flip"：翻转元素到目标的相对一边，再次运行 collision 检测一遍查看元素是否适合。无论哪一边允许更多的元素可见，则使用那一边。
    - "fit"：把元素从窗口的边缘移开。
    - "flipfit"：首先应用 flip 逻辑，把元素放置在允许更多元素可见的那一边。然后应用 fit 逻辑，确保尽可能多的元素可见。

    - "none"：不应用任何 collision 检测。
- **using**（默认值： `null` ） 类型：Function() 描述：当指定了该选项，实际属性设置则委托给该回调。接受两个参数：第一个是位置 top 和 left 值的哈希，可被转发到 .css() 或 .animate()；第二个提供了关于两个元素的位置和尺寸的反馈，同时也计算它们的相对位置。target 和 element 都有下列属性：element、left、top、width、height。另外，还有 horizontal、vertical 和 important，提供了十二个可能的方向，如 { horizontal: "center", vertical: "left", important: "horizontal" }。
- **within**（默认值： `window` ） 类型：Selector 或 Element 或 jQuery 描述：元素定位为 within，会影响 collision 检测。如果您提供了一个选择器（Selector）或 jQuery 对象，则使用第一个匹配的元素。

jQuery UI `.position()` 方法允许您相对窗体（window）、文档、另一个元素或指针（cursor）/鼠标（mouse）来定位一个元素，无需考虑相对父元素的偏移（offset）。

注释：jQuery UI 不支持定位隐藏元素。

这是一个独立的 jQuery 插件，且对其他 jQuery UI 组件没有依赖关系。

该插件扩展自 jQuery 内置的 .position() 方法。如果 jQuery UI 未加载，调用 `.position()` 方法不会直接失败，因为该方法在 jQuery 中存在。但是不会发生预期的行为。

实例

一个简单的 jQuery UI 定位（Position）实例。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>.position() 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  .positionDiv {
    position: absolute;
    width: 75px;
    height: 75px;
    background: green;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="targetElement">
  <div class="positionDiv" id="position1"></div>
  <div class="positionDiv" id="position2"></div>
```

```
  <div class="positionDiv" id="position3"></div>
  <div class="positionDiv" id="position4"></div>
</div>

<script>
$( "#position1" ).position({
  my: "center",
  at: "center",
  of: "#targetElement"
});

$( "#position2" ).position({
  my: "left top",
  at: "left top",
  of: "#targetElement"
});

$( "#position3" ).position({
  my: "right center",
  at: "right bottom",
  of: "#targetElement"
});

$( document ).mousemove(function( event ) {
  $( "#position4" ).position({
    my: "left+3 bottom-3",
    of: event,
    collision: "fit"
  });
});
</script>

</body>
</html>
```

jQuery UI API 类别 - 方法（Methods）

尽管 jQuery UI 主要由 小部件（Widgets）、交互（Interactions） 和 特效（Effects） 组成，但是还有一些简单的方法，增加了便利性。

API	描述	也属于类别
.disableSelection()	禁用选择匹配的元素集合内的文本内容。	UI 核心（UI Core）
.effect()	对一个元素应用动画特效。	特效（Effects） | 特效核心（Effects Core）
.enableSelection()	启用选择匹配的元素集合内的文本内容。	UI 核心（UI Core）
.focus()	异步聚焦到一个元素。	方法重载（Method Overrides） | UI 核心（UI Core）
.hide()	使用自定义效果来隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）
.position()	相对另一个元素定位一个元素。	方法重载（Method Overrides） | 实用工具（Utilities）
.removeUniqueId()	为匹配的元素集合移除由 .uniqueId() 设置的 Id。	UI 核心（UI Core）
.scrollParent()	获取最近的可滚动的祖先。	UI 核心（UI Core）
.show()	使用自定义效果来显示匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）
.toggle()	使用自定义效果来显示或隐藏匹配的元素。	特效（Effects） | 特效核心（Effects Core） | 方法重载（Method Overrides）
.uniqueId()	为匹配的元素集合生成并申请一个唯一的 Id。	UI 核心（UI Core）
.zIndex()	为元素获取 z-index。	UI 核心（UI Core）

jQuery UI API - .disableSelection()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：禁用选择匹配的元素集合内的文本内容。

返回：jQuery

版本新增：1.6

版本废弃：1.9

```
.disableSelection()
```

该方法不接受任何参数。

禁用的文本选择是有害的，不建议使用。

jQuery UI API - .enableSelection()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：启用选择匹配的元素集合内的文本内容。

返回：jQuery

版本新增：1.6

版本废弃：1.9

```
.enableSelection()
```

该方法不接受任何参数。

`.enableSelection()` 方法可用于启用通过 `[.disableSelection()](api-disableSelection.html)` 禁用的文本选择。

jQuery UI API - .removeUniqueId()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：为匹配的元素集合移除由 .uniqueId() 设置的 Id。

返回：jQuery

版本新增：1.9

```
.removeUniqueId()
```

该方法不接受任何参数。

`.removeUniqueId()` 移除由 `[.uniqueId()](api-uniqueId.html)` 设置的 id。在未使用 `.uniqueId()` 设置 id 的元素上调用 `.removeUniqueId()` 则无影响，即使该元素有一个 id。

jQuery UI API - .scrollParent()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：获取最近的可滚动的祖先。

返回：jQuery

```
.scrollParent()
```

该方法不接受任何参数。

该方法查找最近的可滚动的祖先。换句话说，`.scrollParent()` 查找当前所选元素在其内滚动的元素。

该方法只在包含一个元素的 jQuery 对象上工作。

jQuery UI API - .uniqueId()

所属类别

方法（Methods） | UI 核心（UI Core）

用法

描述：为匹配的元素集合生成并申请一个唯一的 Id。

返回：jQuery

版本新增：1.9

```
.uniqueId()
```

该方法不接受任何参数。

许多小部件需要元素生成唯一的 id。 `.uniqueId()` 会检查元素是否有 id，如果元素没有 id，它将生成一个 id，并设置为该元素的 id。在未检查元素是否具有 id 就调用 `.uniqueId()` 是安全的。当小部件使用后需要清除，如果 id 是通过 `.uniqueId()` 添加的， `[.removeUniqueId()](api-removeUniqueId.html)` 方法将从元素上移除 id，如果 id 不是通过 `.uniqueId()` 添加的，则无影响。 `.removeUniqueId()` 之所以能区分 id，是因为 `.uniqueId()` 生成的 id 带有前缀 `"ui-id-"` 。

jQuery UI API - .zIndex()

所属类别

方法（Methods） | UI 核心（UI Core）

.zIndex() 用法

描述：为元素获取 z-index。

返回：jQuery

```
.zIndex()
```

该方法不接受任何参数。

`.zIndex()` 方法在查找一个元素的 z-index 时非常有用，忽略 z-index 是否是直接设置在元素上还是设置在祖先元素上。为了确定 z-index，该方法会在指定的元素上开始，且会沿着 DOM 查找，直到找到一个带有 z-index 的已定位的元素。如果未找到这样的元素，该方法将返回一个 `0` 值。

该方法假设带有嵌套 z-index 的元素不带有一个 `0` 值的 z-index。例如，给出下面的 DOM，内部元素将被当成不带有 z-index，因为在 Internet Explorer 中无法区分一个 `0` 显式值和无值。

```
<div style="z-index: -10;">
  <div style="z-index: 0;"></div>
</div>
```

.zIndex(zIndex) 用法

描述：为元素设置 z-index。

返回：Integer

```
.zIndex( zIndex )
```

参数	类型	描述
zIndex	Integer	要设置的 z-index。

这相当于 `.css( "zIndex", zIndex )`。

jQuery UI API 类别 - 选择器（Selectors）

API	描述	也属于类别
:data() Selector	选择数据已存储在指定的键下的元素。	UI 核心（UI Core）
:focusable Selector	选择可被聚焦的元素。	UI 核心（UI Core）
:tabbable Selector	选择用户可通过 tab 键聚焦的元素。	UI 核心（UI Core）

jQuery UI API - :data() Selector

所属类别

选择器（Selectors） | UI 核心（UI Core）

用法

描述：选择数据已存储在指定的键下的元素。

```
jQuery( ":data(key)" )
```

参数	描述
key	数据的键。

表达式 `$( "div:data(foo)")` 匹配一个通过 `.data( "foo", value )` 存储数据的 `div` 。

实例

选择带有数据的元素，改变它们的背景颜色。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>:data() Selector 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  div {
    width: 100px;
    height: 100px;
    border: 1px solid #000;
    float: left;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="one"></div>
<div id="two"></div>
<div id="three"></div>

<script>
$( "#one" ).data( "color", "blue" );
$( "#three" ).data( "color", "green" );

$( ":data(color)" ).each(function() {
  var element = $( this );
  element.css( "backgroundColor", element.data( "color" ) );
});
</script>

</body>
</html>
```

jQuery UI API - :focusable Selector

所属类别

选择器（Selectors） | UI 核心（UI Core）

用法

描述：选择可被聚焦的元素。

```
jQuery( ":focusable" )
```

一些元素本身是可聚焦的（focusable），而另一些元素需要显式设置 tab 索引。以上两种情况，为了可聚焦（focusable），元素都必须是可见的。

下面类型的元素如果未被禁用，则是可聚焦的（focusable）：`input`、`select`、`textarea`、`button` 和 `object`。锚如果带有 `href` 或 `tabindex` 属性，则是可聚焦的（focusable）。`area` 元素如果在一个已命名的 map 内，且带有 `href` 属性，且有一个可见的图像使用了该 map，则是可聚焦的（focusable）。所有其他完全基于 `tabindex` 属性和可见度的元素是可聚焦的（focusable）。

注释：带有负的 tab 索引的元素是 `:focusable`，不是 `[:tabbable](api-tabbable-selector.html)`。

实例

选择可聚焦的元素，且用一个红色边框突出显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>:focusable Selector 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  input, a, p {
    border: 1px solid #000;
  }
  div {
    padding: 5px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div><input value="文本输入"></div>
<div><a>不带有 href 的锚</a></div>
<div><a href="#">带有 href 的锚</a></div>
<div><p>不带有 tabindex 的段落</p></div>
<div><p tabindex="1">带有 tabindex 的段落</p></div>

<script>
$( ":focusable" ).css( "border-color", "red" );
</script>

</body>
</html>
```

jQuery UI API - :tabbable Selector

所属类别

选择器（Selectors） | UI 核心（UI Core）

用法

描述：选择用户可通过 tab 键聚焦的元素。

```
jQuery( ":tabbable" )
```

一些元素本身是可通过 tab 键聚焦的（tabbable），而另一些元素需要显式设置一个正的 tab 索引。以上两种情况，为了可通过 tab 键聚焦（tabbable），元素都必须是可见的。

下面类型的元素如果不带有负的 tab 索引且未被禁用，则是可通过 tab 键聚焦的（tabbable）： `input` 、 `select` 、 `textarea` 、 `button` 和 `object` 。锚如果带有 `href` 或正的 `tabindex` 属性，则是可通过 tab 键聚焦的（tabbable）。 `area` 元素如果在一个已命名的 map 内，且带有 `href` 属性，且有一个可见的图像使用了该 map，则是可通过 tab 键聚焦的（tabbable）。所有其他完全基于 `tabindex` 属性和可见度的元素是可通过 tab 键聚焦的（tabbable）。

注释：带有负的 tab 索引的元素是 `[:focusable](api-focusable-selector.html)` ，不是 `:tabbable` 。

实例

选择可通过 tab 键聚焦的元素，且用一个红色边框突出显示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>:tabbable Selector 实例</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  input {
    border: 1px solid #000;
  }
  div {
    padding: 5px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div><input value="没有 tabindex"></div>
<div><input tabindex="5" value="正的 tabindex"></div>
<div><input tabindex="-1" value="负的 tabindex"></div>

<script>
$( ":tabbable" ).css( "border-color", "red" );
</script>

</body>
</html>
```

jQuery UI API 类别 - 主题（Theming）

jQuery UI 包括一个强大的 CSS 框架，用于创建自定义的 jQuery 小部件。该框架包括涵盖广泛的公共用户界面需求的 Class，并且可以使用 jQuery UI ThemeRoller 进行操纵。通过使用 jQuery UI CSS 框架创建您自己的 UI 组件，您应采用共享标记公约，便于插件社区的代码集成。您可以查看 jQuery UI 主题 了解更多详情。

API	描述
CSS 框架（CSS Framework）	jQuery UI 使用的允许组件主题化的 Class 名称列表。
图标（Icons）	jQuery UI 提供的图标列表。
堆叠元素（Stacking Elements）	一种处理 z-index 和堆叠元素的模式。

jQuery UI API - CSS 框架（CSS Framework）

下面是 jQuery UI 使用的 Class 名称列表。这些 Class 用来创建跨应用程序的视觉一致性，且允许组件通过 jQuery UI ThemeRoller 进行主题化。下面的 CSS 类根据样式是否是固定的结构化的，或者是否是可主题化的（颜色、字体、背景等），分别定义在 ui.core.css 和 ui.theme.css 两个文件中。

布局助手

- `.ui-helper-hidden` ：对元素应用 `display: none` 。
- `.ui-helper-hidden-accessible` ：对元素应用访问隐藏（通过页面绝对定位）。
- `.ui-helper-reset` ：UI 元素的基本样式重置。重置的元素比如： `padding` 、 `margin` 、 `text-decoration` 、list-style，等等。
- `.ui-helper-clearfix` ：对父元素应用浮动包装属性。
- `.ui-helper-zfix` ：对 `<iframe>` 元素应用 iframe "fix" CSS。
- `.ui-front` ：应用 z-index 来管理屏幕上多个小部件的堆叠，如需了解更多详情，请查看 堆叠元素（Stacking Elements） 页面。

小部件容器

- `.ui-widget` ：对所有小部件的外部容器应用的 Class。对小部件应用字体和字体尺寸，同时也对自表单元素应用相同的字体和 1em 的字体尺寸，以应对 Windows 浏览器中的继承问题。
- `.ui-widget-header` ：对标题容器应用的 Class。对元素及其子元素的文本、链接、图标应用标题容器样式。
- `.ui-widget-content` ：对内容容器应用的 Class。对元素及其子元素的文本、链接、图标应用内容容器样式。（可应用到标题的父元素或者同级元素）

交互状态

- `.ui-state-default` ：对可点击按钮元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable default" 容器样式。
- `.ui-state-hover` ：当鼠标悬浮在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-focus` ：当键盘聚焦在可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable hover" 容器样式。
- `.ui-state-active` ：当鼠标点击可点击按钮元素上时应用的 Class。对元素及其子元素的文本、链接、图标应用 "clickable active" 容器样式。

交互提示 Cues

- `.ui-state-highlight` ：对高亮或者选中元素应用的 Class。对元素及其子

元素的文本、链接、图标应用 "highlight" 容器样式。
- `.ui-state-error` ：对错误消息容器元素应用的 Class。对元素及其子元素的文本、链接、图标应用 "error" 容器样式。
- `.ui-state-error-text` ：对只有无背景的错误文本颜色应用的 Class。可用于表单标签，也可以对子图标应用错误图标颜色。
- `.ui-state-disabled` ：对禁用的 UI 元素应用一个暗淡的不透明度。意味着对一个已经定义样式的元素添加额外的样式。
- `.ui-priority-primary` ：对第一优先权的按钮应用的 Class。应用粗体文本。
- `.ui-priority-secondary` ：对第二优先权的按钮应用的 Class。应用正常粗细的文本，对元素应用轻微的透明度。

图标

状态和图像

- `.ui-icon` ：对图标元素应用的基本 Class。设置尺寸为 16px 方块，隐藏内部文本，对 "content" 状态的精灵图像设置背景图像。注意： *`.ui-icon` class* 将根据它的父容器得到一个不同的精灵背景图像。例如， *`ui-state-default`* 容器内的 *`ui-icon`* 元素将根据 *`ui-state-default`* 的图标颜色进行着色。

图标类型

在声明 `.ui-icon` class 之后，接着您可以声明一个秒速图标类型的 class。通常情况下，图标 class 遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}` 。

例如，一个指向右侧的三角形图标，如下所示： `.ui-icon-triangle-1-e`

jQuery UI 的 ThemeRoller 在它的预览一栏中提供了全套的 CSS 框架图标。将鼠标悬浮在图标上可查看 class 名称。

其他视觉效果

圆角半径助手

- `.ui-corner-tl` ：对元素的左上角应用圆角半径。
- `.ui-corner-tr` ：对元素的右上角应用圆角半径。
- `.ui-corner-bl` ：对元素的左下角应用圆角半径。
- `.ui-corner-br` ：对元素的右下角应用圆角半径。
- `.ui-corner-top` ：对元素上边的左右角应用圆角半径。
- `.ui-corner-bottom` ：对元素下边的左右角应用圆角半径。
- `.ui-corner-right` ：对元素右边的上下角应用圆角半径。
- `.ui-corner-left` ：对元素左边的上下角应用圆角半径。

- `.ui-corner-all` ：对元素的所有四个角应用圆角半径。

覆盖 & 阴影

- `.ui-widget-overlay` ：对覆盖屏幕应用 100% 宽度和高度，同时设置背景颜色/纹理和屏幕不透明度。
- `.ui-widget-shadow` ：对覆盖应用的 Class，设置了不透明度、上偏移/左偏移，以及阴影的 "厚度"。厚度是通过对阴影所有边设置内边距（padding）进行应用的。偏移是通过设置上外边距（margin）和左外边距（margin）进行应用的（可以是正数，也可以是负数）。

jQuery UI API - 图标（Icons）

jQuery UI 提供了大量可以通过对元素应用 Class 名称来使用的图标（Icons）。Class 名称大体上遵循语法 `.ui-icon-{icon type}-{icon sub description}-{direction}`。例如，下面将显示一个向北的厚箭头的图标：

```
<span class="ui-icon ui-icon-arrowthick-1-n"></span>
```

图标也集成到一些 jQuery UI 的小部件，比如 accordion、button、menu。

下面是 jQuery UI 提供的图标的完整列表：

jQuery UI API - 堆叠元素（Stacking Elements）

堆叠的或者移动到其他元素前面的小部件（Widgets）当放置到现实世界的页面中时经常面临挑战。通常通过简单地改变堆叠元素的 `z-index` 或者父元素来避免页面上的冲突。但是，jQuery UI 需要一个不需要手动改变 `z-index` 值的通用的解决方案。这是通过 `ui-front` class 来完成的，通常还伴随着堆叠组件上的 `appendTo` 选项。

`ui-front` class

`ui-front` class 是非常基础的。它只是在元素上设置了一个静态的 `z-index` 值。但是，class 的存在是用来表明堆叠元素要追加到哪里。这允许我们利用嵌套层内容，生成一个在大多数情况下都能使用的默认的 DOM 位置。

注释：当使用 *`ui-front`* 时，您必须设置 *`position`* 为 *`relative`*、*`absolute`* 或 *`fixed`*，以便应用 *`z-index`*。

堆叠技术（stacking technique）

追加堆叠元素到页面的任何小部件都必须使用 `ui-front` class，且在大多数情况下，应该有一个 `appendTo` 选项。堆叠元素应遵循下面的规则：

- 如果一个值设置为 `appendTo` 选项，则追加堆叠元素到指定的元素。
- 如果 `appendTo` 选项被设置为 `null`（默认），则小部件应从相关的元素开始遍历 DOM。例如，当自动完成（autocomplete）菜单被追加到 DOM，遍历则从相关的 input 元素开始。
    - 如果找到一个带有 `ui-front` class 的元素，则追加到该元素。
    - 如果没有找到一个带有 `ui-front` class 的元素，则追加到主体（body）。
- 堆叠元素的 `position` 必须设置为 `relative`、`absolute` 或 `fixed`，以便应用来自 `ui-front` class 的 `z-index`。使用 .position() 将自动设置 `position`。

jQuery UI API 类别 - UI 核心（UI Core）

由 jquery.ui.core.js 提供的功能。

API	描述	也属于类别
:data() Selector	选择数据已存储在指定的键下的元素。	选择器（Selectors）
.disableSelection()	禁用选择匹配的元素集合内的文本内容。	方法（Methods）
.enableSelection()	启用选择匹配的元素集合内的文本内容。	方法（Methods）
.focus()	异步聚焦到一个元素。	方法重载（Method Overrides） | 方法（Methods）
:focusable Selector	选择可被聚焦的元素。	选择器（Selectors）
jQuery.ui.keyCode	一个相对于数字值的关键代码描述的映射。	
.removeUniqueId()	为匹配的元素集合移除由 .uniqueId() 设置的 Id。	方法（Methods）
.scrollParent()	获取最近的可滚动的祖先。	方法（Methods）
:tabbable Selector	选择用户可通过 tab 键聚焦的元素。	选择器（Selectors）
.uniqueId()	为匹配的元素集合生成并申请一个唯一的 Id。	方法（Methods）
.zIndex()	为元素获取 z-index。	方法（Methods）

jQuery UI API 类别 - 实用工具（Utilities）

API	描述	也属于类别
Easings	Easing 函数指定动画在不同点上的行进速度。	
部件库（Widget Factory）	使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。	小部件（Widgets）
插件桥（Widget Plugin Bridge）	jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。	小部件（Widgets）
鼠标交互（Mouse Interaction）	基本交互层。	交互（Interactions）
.position()	相对另一个元素定位一个元素。	方法重载（Method Overrides） | 方法（Method）

jQuery UI API - 部件库（Widget Factory）

所属类别

实用工具（Utilities） | 小部件（Widgets）

jQuery.widget(name [, base], prototype) 用法

描述：使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。

```
jQuery.widget( name [, base ], prototype )
```

参数	类型	类型
name	String	要创建的小部件名称，包括命名空间。
base	Function()	要继承的基础小部件。必须是一个可以使用 new 关键词实例化的构造函数。默认为 jQuery.Widget。
prototype	PlainObject	要作为小部件原型使用的对象。

您可以使用 `$.Widget` 对象作为要继承的基础，或者可以明确地从现有的 jQuery UI 或第三方控件，从头开始创建新的小部件。定义一个带有相同名称的小部件来继承基础部件，甚至允许您适当地扩展小部件。

jQuery UI 中包含许多保持状态的小部件，因此比典型的 jQuery 插件稍有不同的使用模式。所有的jQuery UI 小部件使用相同的模式，这是由部件库（Widget Factory）定义的。所以，只要您学会使用其中一个，您就知道如何使用其他的小部件（Widget）。

注释：本章节使用 进度条部件（Progressbar Widget） 演示实例，但是语法适用于每个小部件。

初始化

为了跟踪小部件的状态，我们必须引入小部件的全生命周期。小部件初始化时生命周期开始。要初始化一个小部件，我们只需要简单地在一个或多个元素上调用插件。

```
$( "#elem" ).progressbar();
```

这将初始化 jQuery 对象中的每个元素。上面实例中元素 id 为 `"elem"`。

选项

由于 `progressbar()` 调用时不带参数，小部件是使用默认选项进行初始化的。我们可以在初始化时传递一组选项来覆盖默认选项：

```
$( "#elem" ).progressbar({ value: 20 });
```

我们可以根据需要传递选项的个数，任何我们未传递的选项都使用它们的默认值。

您可以传递多个选项参数，这些参数将会被合并为一个对象（类似于 `$.extend( true, target, object1, objectN )`）。这在为所有实例覆盖一些设置，实例间共享选项时很有用：

```
var options = { modal: true, show: "slow" };
$( "#dialog1" ).dialog( options );
$( "#dialog2" ).dialog( options, { autoOpen: false });
```

所有在初始化时传递的选项都是深拷贝的，确保后续在不影响小部件的情况下修改对象。数组是唯一的例外，它们是按原样引用的。这个例外是为了适当地支持数据绑定，其中数据源必须作为引用。

默认值保存在小部件的属性中，因此我们可以覆盖 jQuery UI 设置的值。例如，在下面的设置后，所有将来的进度条实例将默认为值 80：

```
$.ui.progressbar.prototype.options.value = 80;
```

选项是小部件状态的组成部分，所以我们也可以在初始化后设置选项。我们会在后续看到 option 方法。

方法

现在小部件已经初始化，我们可以查询它的状态，或者在小部件上执行动作。所有初始化后的动作都是以方法调用方式执行。为了在小部件上调用一个方法，我们向 jQuery 插件传递方法的名称。例如，在进度条部件（Progressbar Widget）上调用 `value()` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value" );
```

如果方法接受参数，我们可以在方法名称后传递参数。例如，要传递参数 `40` 到 `value()` 方法，我们可以使用：

```
$( "#elem" ).progressbar( "value", 40 );
```

就像 jQuery 中的其他方法，大多数的小部件方法返回 jQuery 对象：

```
$( "#elem" )
  .progressbar( "value", 90 )
  .addClass( "almost-done" );
```

每个小部件都有自己的方法设置，这些设置是基于小部件提供的功能。但是，有一些方法是存在于所有的小部件上，这会在下面进行详细讲解。

事件

所有的小部件都有与它们各种行为相关的事件，以便在状态改变的时候通知您。对于大多数的小部件，当事件被触发时，名称以小部件名称的小写字母形式作为前缀。例如，我们可以绑定进度条的 `change` 事件，该事件在值改变时触发。

```
$( "#elem" ).bind( "progressbarchange", function() {
  alert( "The value has changed!" );
});
```

每个事件都有一个对应的回调，这会作为选项。如果需要，我们可以抓住进度条的 `change` 回调，而不用绑定 `progressbarchange` 事件。

```
$( "#elem" ).progressbar({
  change: function() {
    alert( "The value has changed!" );
  }
});
```

所有的小部件都有一个 `change` 事件，该事件在实例化时触发。

实例化

小部件的实例是使用带有小部件全称作为键的 `jQuery.data()` 存储的。因此，您可以使用下面代码从元素检索进度条部件（Progressbar Widget）的实例对象。

```
$( "#elem" ).data( "ui-progressbar" );
```

元素是否绑定了给定小部件，可以使用 `:data` 选择器来检测。

```
$( "#elem" ).is( ":data( 'ui-progressbar' )" ); // true
$( "#elem" ).is( ":data( 'ui-draggable' )" ); // false
```

您也可以使用 `:data` 来获得作为给定小部件实例的所有元素的列表。

```
$( ":data( 'ui-progressbar' )" );
```

属性

所有的小部件都有下面的属性：

- **defaultElement**：当构造小部件实例未提供元素时要使用的元素。例如，由于进度条的 `defaultElement` 是 `"<div> "`， `$.ui.progressbar({ value: 50 })` 在一个新建的 `<div>` 上实例化进度条小部件实例。
- **document**：其内包含小部件元素的 `document` 。如果需要在框架内与小部件进行交互时很有用。
- **element**：一个 jQuery 对象，包含用于实例化小部件的 元素。如果您选择多个元素并调用 `.myWidget()` ，将为每个元素创建一个单独的小部件实例。因此，该属性总是包含一个元素。
- **namespace**：小部件原型存储的全局 jQuery 对象的位置。例如， `"ui"` 的 `namespace` 表示小部件原型存储在 `$.ui` 。
- **options**：一个包含小部件当前使用选项的对象。在实例化时，用户提供的任何选项将会自动与 `$.myNamespace.myWidget.prototype.options` 中定义的默认值合并。用户指定的选项会覆盖默认值。
- **uuid**：一个表示控件标识符的唯一整数。
- **version**：小部件的字符串版本。对于 jQuery UI 小部件，该属性会被设置为小部件使用的 jQuery UI 的版本。插件开发者必须在他们的原型中明确设置该属性。
- **widgetEventPrefix**：添加到小部件事件名称的前缀。例如，可拖拽小部件（Draggable Widget）的 `widgetEventPrefix` 是 `"drag"` ，因此当创建一个 draggable 时，事件的名称是 `"dragcreate"` 。默认情况下，小部件的 `widgetEventPrefix` 是它的名称。注意：该属性已被废弃，将在以后的版本中非常。事件名称将被改为 *widgetName:eventName* （例如 *`"draggable:create"`* ）。
- **widgetFullName**：包含命名空间的小部件全称。对于 `$.widget( "myNamespace.myWidget", {} )` ， `widgetFullName` 将是 `"myNamespace-myWidget"` 。
- **widgetName**：小部件的名称。对于 `$.widget( "myNamespace.myWidget", {} )` ， `widgetName` 将是 `"myWidget"` 。
- **window**：其内包含小部件元素的 `window` 。如果需要在框架内与小部件进行交互时很有用。

jQuery.Widget 基础小部件用法

描述：部件库（Widget Factory）使用的基础小部件。

快速导航

选项	
disabledhideshow	_create_delay_destroy_focusable_getCreateEventData_ge

disabled

类型：Boolean

描述：如果设置为 `true` ，则禁用该小部件。

代码实例：

初始化带有指定 `disabled` 选项的小部件：

```
$( ".selector" ).widget({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).widget( "option", "disabled" );

// setter
$( ".selector" ).widget( "option", "disabled", true );
```

默认值：false

hide

类型：Boolean 或 Number 或 String 或 Object

描述：是否使用动画隐藏元素，以及如何动画隐藏元素。

支持多个类型：

- **Boolean**：当设置为 `false` 时，则不使用动画，元素会立即隐藏。当设置为 `true` 时，元素会使用默认的持续时间和默认的 easing 淡出。
- **Number**：元素将使用指定的持续时间和默认的 easing 淡出。
- **String**：元素将使用指定的特效隐藏。该值可以是一个内建的 jQuery 动画方法名称，比如 `"slideUp"` ，也可以是一个 jQuery UI 特效 的名称，比如 `"fold"` 。以上两种情况的特效将使用默认的持续时间和默认的 easing。
- **Object**：如果值是一个对象，则 `effect` 、 `delay` 、 `duration` 和 `easing` 属性会被提供。如果 `effect` 属性包含 jQuery 方法的名称，则使用该方法，否则，它被认为是一个 jQuery UI 特效的名称。当使用一个支持额外设置的 jQuery UI 特效时，您可以在对象中包含这些设置，且它们会被传递

给特效。如果 `duration` 或 `easing` 被省略，则使用默认值。如果 `effect` 被省略，则使用 `"fadeOut"` 。如果 `delay` 被省略，则不使用延迟。

代码实例：

初始化带有指定 `hide` 选项的小部件：

```
$( ".selector" ).widget({ hide: { effect: "explode", duration: 100(
```

在初始化后，获取或设置 `hide` 选项：

```
// getter
var hide = $( ".selector" ).widget( "option", "hide" );

// setter
$( ".selector" ).widget( "option", "hide", { effect: "explode", du
```

默认值：null

show

类型：Boolean 或 Number 或 String 或 Object

描述：是否使用动画显示元素，以及如何动画显示元素。

支持多个类型：

- **Boolean**：当设置为 `false` 时，则不使用动画，元素会立即显示。当设置为 `true` 时，元素会使用默认的持续时间和默认的 easing 淡入。
- **Number**：元素将使用指定的持续时间和默认的 easing 淡入。
- **String**：元素将使用指定的特效显示。该值可以是一个内建的 jQuery 动画方法名称，比如 `"slideDown"` ，也可以是一个 jQuery UI 特效 的名称，比如 `"fold"` 。以上两种情况的特效将使用默认的持续时间和默认的 easing。
- **Object**：如果值是一个对象，则 `effect` 、 `delay` 、 `duration` 和 `easing` 属性会被提供。如果 `effect` 属性包含 jQuery 方法的名称，则使用该方法，否则，它被认为是一个 jQuery UI 特效的名称。当使用一个支持额外设置的 jQuery UI 特效时，您可以在对象中包含这些设置，且它们会被传递给特效。如果 `duration` 或 `easing` 被省略，则使用默认值。如果 `effect` 被省略，则使用 `"fadeIn"` 。如果 `delay` 被省略，则不使用延迟。

代码实例：

初始化带有指定 `show` 选项的小部件：

```
$( ".selector" ).widget({ show: { effect: "blind", duration: 800 }
```

在初始化后，获取或设置 `show` 选项：

```
// getter
var show = $( ".selector" ).widget( "option", "show" );

// setter
$( ".selector" ).widget( "option", "show", { effect: "blind", durat
```

默认值：null

方法

_create()

类型：jQuery (plugin only)

描述：`_create()` 方法是小部件的构造函数。没有参数，但是 `this.element` 和 `this.options` 已经设置。

- 该方法不接受任何参数。

代码实例：

基于一个选项设置小部件元素的背景颜色。

```
_create: function() {
  this.element.css( "background-color", this.options.color );
}
```

_delay(fn [, delay])

类型：Number

描述：在指定延迟后调用提供的函数。保持 `this` 上下文正确。本质上是 `setTimeout()`。

使用 `clearTimeout()` 返回超时 ID。

- **fn** 类型：Function() 或 String 描述：要调用的函数。也可以是小部件上方法的名称。
- **delay** 类型：Number 描述：调用函数前等待的毫秒数，默认为 `0`。

代码实例：

100 毫秒后在小部件上调用 `_foo()` 方法。

```
this._delay( this._foo, 100 );
```

_destroy()

类型：jQuery (plugin only)

描述：公共的 `destroy()` 方法清除所有的公共数据、事件等等。代表了定制、指定小部件、清理的 `_destroy()` 。

- 该方法不接受任何参数。

代码实例：

当小部件被销毁时，从小部件的元素移除一个 class。

```
_destroy: function() {
  this.element.removeClass( "my-widget" );
}
```

_focusable(element)

类型：jQuery (plugin only)

描述：建立聚焦在元素上时要应用 `ui-state-focus` class 的 `element` 。

- **element** 类型：jQuery 描述：要应用 focusable 行为的元素。

代码实例：

向小部件内的一组元素应用 focusable 样式：

```
this._focusable( this.element.find( ".my-items" ) );
```

_getCreateEventData()

类型：Object

描述：所有的小部件触发 `create` 事件。默认情况下，事件中不提供任何的数据，但是该方法会返回一个对象，作为 `create` 事件的数据被传递。

- 该方法不接受任何参数。

代码实例：

向 `create` 事件处理程序传递小部件的选项，作为参数。

```
_getCreateEventData: function() {
  return this.options;
}
```

_getCreateOptions()

类型：Object

描述：该方法允许小部件在初始化期间为定义选项定义一个自定义的方法。用户提供的选项会覆盖该方法返回的选项，即会覆盖默认的选项。

- 该方法不接受任何参数。

代码实例：

让小部件元素的 id 属性作为选项可用。

```
_getCreateOptions: function() {
  return { id: this.element.attr( "id" ) };
}
```

_hide(element, option [, callback])

类型：jQuery (plugin only)

描述：使用内置的动画方法或使用自定义的特效隐藏一个元素。如需了解可能的 `option` 值，请查看 hide。

- **element** 类型：jQuery 描述：要隐藏的元素。
- **option** 类型：Object 描述：定义如何隐藏元素的设置。
- **callback** 类型：Function() 描述：元素完全隐藏后要调用的回调。

代码实例：

为自定义动画传递 `hide` 选项。

```
this._hide( this.element, this.options.hide, function() {

  // Remove the element from the DOM when it's fully hidden.
  $( this ).remove();
});
```

_hoverable(element)

类型：jQuery (plugin only)

描述：建立悬浮在元素上时要应用 `ui-state-hover` class 的 `element` 。事件处理程序在销毁时自动清理。

- **element** 类型：jQuery 描述：要应用 hoverable 行为的元素。

代码实例：

当悬浮在元素上时，向元素内所有的 `div` 应用 hoverable 样式。

```
this._hoverable( this.element.find( "div" ) );
```

_init()

类型：jQuery (plugin only)

描述：小部件初始化的理念与创建不同。任何时候不带参数的调用插件或者只带一个选项哈希的调用插件，初始化小部件。当小部件被创建时会包含这个方法。

注释：如果存在不带参数成功调用小部件时要执行的逻辑动作，初始化只能在这时处理。

- 该方法不接受任何参数。

代码实例：

如果设置了 `autoOpen` 选项，则调用 `open()` 方法。

```
_init: function() {
  if ( this.options.autoOpen ) {
    this.open();
  }
}
```

_off(element, eventName)

类型：jQuery (plugin only)

描述：从指定的元素取消绑定事件处理程序。

- **element** 类型：jQuery 描述：要取消绑定事件处理程序的元素。不像 `_on()` 方法，`_off()` 方法中元素是必需的。
- **eventName** 类型：String 描述：一个或多个空格分隔的事件类型。

代码实例：

从小部件的元素上取消绑定所有 click 事件。

```
this._off( this.element, "click" );
```

_on([suppressDisabledCheck] [, element], handlers)

类型：jQuery (plugin only)

描述：授权通过事件名称内的选择器被支持，例如 `"click .foo"` 。 `_on()` 方法提供了一些直接事件绑定的好处：

- 保持处理程序内适当的 `this` 上下文。
- 自动处理禁用的部件：如果小部件被禁用或事件发生在带有 `ui-state-disabled` class 的元素上，则不调用事件处理程序。可以被 `suppressDisabledCheck` 参数重写。
- 事件处理程序会自动添加命名空间，在销毁时会自自动清理。
- **suppressDisabledCheck**（默认值： `false` ） 类型：Boolean 描述：是否要绕过禁用的检查。
- **element** 类型：jQuery 描述：要绑定事件处理程序的元素。如果未提供元素， `this.element` 用于未授权的事件，widget 元素 用于授权的事件。
- **handlers** 类型：Object 描述：一个 map，其中字符串键代表事件类型，可选的选择器用于授权，值代表事件调用的处理函数。

代码实例：

放置小部件元素内所有被点击的链接的默认行为。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
  }
});
```

_setOption(key, value)

类型：jQuery (plugin only)

描述：为每个独立的选项调用 `_setOptions()` 方法。小部件状态随着改变而更新。

- **key** 类型：String 描述：要设置的选项名称。
- **value** 类型：Object 描述：为选项设置的值。

代码实例：

当小部件的 `height` 或 `width` 选项改变时，更新小部件的元素。

```
_setOption: function( key, value ) {
  if ( key === "width" ) {
    this.element.width( value );
  }
  if ( key === "height" ) {
    this.element.height( value );
  }
  this._super( key, value );
}
```

_setOptions(options)

类型：jQuery (plugin only)

描述：当调用 `option()` 方法时调用，无论以什么形式调用 `option()` 。如果您要根据多个选项的改变而改变处理器密集型，重载该方法是很有用的。

- **options** 类型：Object 描述：为选项设置的值。

代码实例：

如果小部件的 `height` 或 `width` 选项改变，调用 `resize` 方法。

```
_setOptions: function( options ) {
  var that = this,
    resize = false;

  $.each( options, function( key, value ) {
    that._setOption( key, value );
    if ( key === "height" &#124;&#124; key === "width" ) {
      resize = true;
    }
  });

  if ( resize ) {
    this.resize();
  }
}
```

_show(element, option [, callback])

类型：jQuery (plugin only)

描述：使用内置的动画方法或使用自定义的特效显示一个元素。如需了解可能的 `option` 值，请查看 show。

- **element** 类型：jQuery 描述：要显示的元素。
- **option** 类型：Object 描述：定义如何显示元素的设置。

- **callback** 类型：Function() 描述：元素完全显示后要调用的回调。

代码实例：

为自定义动画传递 `show` 选项。

```
this._show( this.element, this.options.show, function() {

  // Focus the element when it's fully visible.
  this.focus();
}
```

_super([arg] [, ...])

类型：jQuery (plugin only)

描述：从父部件中调用相同名称的方法，带有任意指定的参数。本质上是 `.call()` 。

- **arg** 类型：Object 描述：要传递给父部件的方法的零到多个参数。

代码实例：

处理 `title` 选项更新，并调用付部件的 `_setOption()` 来更新选项的内部存储。

```
_setOption: function( key, value ) {
  if ( key === "title" ) {
    this.element.find( "h3" ).text( value );
  }
  this._super( key, value );
}
```

_superApply(arguments)

类型：jQuery (plugin only)

描述：从父部件中调用相同名称的方法，带有参数的数组。本质上是 `.apply()` 。

- **arguments** 类型：Array 描述：要传递给父部件的方法的参数数组。

代码实例：

处理 `title` 选项更新，并调用付部件的 `_setOption()` 来更新选项的内部存储。

```
_setOption: function( key, value ) {
  if ( key === "title" ) {
    this.element.find( "h3" ).text( value );
  }
  this._superApply( arguments );
}
```

_trigger(type [, event] [, data])

类型：Boolean

描述：触发一个事件及其相关的回调。带有该名称的选项与作为回调被调用的类型相等。

事件名称是小部件名称和类型的小写字母串。

注释：当提供数据时，您必须提供所有三个参数。如果没有传递事件，则传递 `null` 。

如果默认行为是阻止的，则返回 `false` ，否则返回 `true` 。当处理程序返回 `false` 时或调用 `event.preventDefault()` 时，则阻止默认行为发生。

- **type** 类型：String 描述： `type` 应该匹配回调选项的名称。完整的事件类型会自动生成。
- **event** 类型：Event 描述：导致该事件发生的原始事件，想听众提供上下文时很有用。
- **data** 类型：Object 描述：一个与事件相关的数据哈希。

代码实例：

当按下一个键时，触发 `search` 事件。

```
this._on( this.element, {
  keydown: function( event ) {

    // Pass the original event so that the custom search event has
    // useful information, such as keyCode
    this._trigger( "search", event, {

      // Pass additional information unique to this event
      value: this.element.val()
    });
  }
});
```

destroy()

类型：jQuery (plugin only)

描述：完全移除小部件功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

当点击小部件的任意锚点时销毁小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.destroy();
  }
});
```

disable()

类型：jQuery (plugin only)

描述：禁用小部件。

- 该方法不接受任何参数。

代码实例：

当点击小部件的任意锚点时禁用小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.disable();
  }
});
```

enable()

类型：jQuery (plugin only)

描述：启用小部件。

- 该方法不接受任何参数。

代码实例：

当点击小部件的任意锚点时启用小部件。

```
this._on( this.element, {
  "click a": function( event ) {
    event.preventDefault();
    this.enable();
  }
});
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

获得 `width` 选项的值。

```
this.option( "width" );
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前小部件选项哈希。

- 该方法不接受任何参数。

代码实例：

记录每个小部件选项的键/值对，用于调试。

```
var options = this.option();
for ( var key in options ) {
  console.log( key, options[ key ] );
}
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的小部件选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

设置 `width` 选项为 `500`。

```
this.option( "width", 500 );
```

option(options)

类型：jQuery (plugin only)

描述：为小部件设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

设置 `height` 和 `width` 选项为 `500`。

```
this.option({
  width: 500,
  height: 500
});
```

widget()

类型：jQuery

描述：返回一个包含原始元素或其他相关的生成元素的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

当创建小部件时，在小部件的原始元素周围放置一个红色的边框。

```
_create: function() {
  this.widget().css( "border", "2px solid red" );
}
```

事件

create(event, ui)

类型：widgetcreate

描述：当小部件被创建时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的小部件：

```
$( ".selector" ).widget({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 widgetcreate 事件：

```
$( ".selector" ).on( "widgetcreate", function( event, ui ) {} );
```

jQuery UI API - 插件桥（Widget Plugin Bridge）

所属类别

实用工具（Utilities） | 小部件（Widgets）

用法

描述：jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory） 的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。

```
jQuery.widget.bridge( name, constructor )
```

参数	类型	类型
name	String	要创建的插件名称。
constructor	Function()	当插件被调用时要实例化的对象。

`$.widget.bridge()` 做如下事情：

- 连接一个常规的 JavaScript 构造函数到 jQuery API。
- 自动创建对象实例，并存储在元素的 `$.data` 缓存内。
- 允许调用公有方法。
- 防止调用私有方法。
- 防止在未初始化的对象上调用方法。
- 防止多个初始化。

jQuery UI 小部件使用 `$.widget( "foo.bar", {} );` 语法定义一个对象来创建。给出一个带有五个 `.foo` ， `$( ".foo" ).bar();` 的 DOM 结构将创建 "bar" 对象的五个实例。 `$.widget.bridge()` 基于 "bar" 对象和一个公共的 API 在库内工作。因此，您可以通过编写 `$( ".foo" ).bar()` 来创建实例，通过编写 `$( ".foo" ).bar( "baz" )` 来调用方法。

如果您只想一次性初始化并调用方法，那么您所传递给 `jQuery.widget.bridge()` 的对象可以很小：

```
var Highlighter = function( options, element ) {
  this.options = options;
  this.element = $( element );
  this._set( 800 );
};
Highlighter.prototype = {
  toggle: function() {
    this._set( this.element.css( "font-weight") === 400 ? 800 : 400
  },
  _set: function(value) {
    this.element.css( "font-weight", value );
  }
};
```

在这里，您需要的只是一个构造函数，接收两个参数：

- `options` ：一个配置选项的对象
- `element` ：该实例在其上创建的 DOM 元素

然后您可以使用桥（bridge）把该对象作为一个 jQuery 插件，且可以在任意的 jQuery 对象上使用它：

```
// Hook up the plugin
$.widget.bridge( "colorToggle", Highlighter );

// Initialize it on divs
$( "div" ).colorToggle().click(function() {
  // Call the public method on click
  $( this ).colorToggle( "toggle" );
});
```

为了使用桥（bridge）的所有特性，您的对象原型需要有一个 `_init()` 方法。该方法在调用插件且实例已存在时调用。在这种情况下，您还需要有一个 `option()` 方法。该方法将会以选项作为第一个参数被调用。如果没有选项，则参数为一个空对象。如需了解 `option` 方法的使用，请查看 `$.Widget` 。

桥（bridge）有一个可选的属性，如果存在：如果对象原型有一个 `widgetFullName` 属性，则该属性将被作为存储和检索实例的键。否则，将使用 name 参数。

jQuery UI API 类别 - 小部件（Widgets）

小部件（Widgets）是功能丰富、有状态的插件，它有一个完整的生命周期，带有方法和事件。您可以查看 部件库（Widget Factory）文档 了解更多详情。

API	描述	也属于类别
折叠面板部件（Accordion Widget）	把一对标题和内容面板转换成折叠面板。	
自动完成部件（Autocomplete Widget）	自动完成功能根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。	
按钮部件（Button Widget）	可主题化的按钮和按钮集合。	
日期选择器部件（Datepicker Widget）	从弹出框或在线日历选择一个日期。	
对话框部件（Dialog Widget）	在一个交互覆盖层中打开内容。	
部件库（Widget Factory）	使用与所有 jQuery UI 小部件相同的抽象化来创建有状态的 jQuery 插件。	实用工具（Utilities）
插件桥（Widget Plugin Bridge）	jQuery.widget.bridge() 方法是 jQuery 部件库（Widget Factory）的一部分。它扮演着由 $.widget() 创建的对象和 jQuery API 之间的中介。	实用工具（Utilities）
菜单部件（Menu Widget）	带有鼠标和键盘交互的用于导航的可主题化菜单。	
进度条部件（Progressbar Widget）	显示一个确定的或不确定的进程状态。	
滑块部件（Slider Widget）	拖动手柄可以选择一个数值。	
旋转器部件（Spinner	通过向上/向下按钮和箭头键处理，为输入数	

标签页部件（Tabs Widget）	一种多面板的单内容区，每个面板与列表中的标题相关。
工具提示框部件（Tooltip Widget）	可自定义的、可主题化的工具提示框，替代原生的工具提示框。

jQuery UI API - 折叠面板部件（Accordion Widget）

所属类别

小部件（Widgets）

用法

描述：把一对标题和内容面板转换成折叠面板。

版本新增：1.0

折叠面板容器的标记需要一对标题和内容面板。

```
<div id="accordion">
  <h3>First header</h3>
  <div>First content panel</div>
  <h3>Second header</h3>
  <div>Second content panel</div>
</div>
```

折叠面板支持任意标记，但是每个内容面板必须是与其相关的头部面板的下一个同级。请查看 `header` 选项了解如何使用自定义的标记结构。

面板可以通过设置 `active` 选项以编程的方式激活。

键盘交互

当焦点在标题（header）上时，下面的键盘命令可用：

- UP/LEFT - 移动焦点到上一个标题（header）。如果在第一个标题（header）上，则移动焦点到最后一个标题（header）上。
- DOWN/RIGHT - 移动焦点到下一个标题（header）。如果在最后一个标题（header）上，则移动焦点到第一个标题（header）上。
- HOME - 移动焦点到第一个标题（header）上。
- END - 移动焦点到最后一个标题（header）上。
- SPACE/ENTER - 激活与获得焦点的标题（header）相关的面板（panel）。

当焦点在面板（panel）中时，下面的键盘命令可用：

- CTRL+UP：移动焦点到相关的标题（header）。

主题化

折叠面板部件（Accordion Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用折叠面板指定的样式，则可以使用下面的 CSS class 名称：

- `ui-accordion` ：折叠面板的外层容器。
    - `ui-accordion-header` ：折叠面板的标题。如果标题包含 `icons` ，则标题会另外有个 `ui-accordion-icons` class。
    - `ui-accordion-content` ：折叠面板的内容面板。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 特效核心（Effects Core） （可选的；当与 `animate` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

选项	
activeanimatecollapsibledisabledeventheaderheightStyleicons	destroydisablee

active

类型：Boolean 或 Integer

描述：当前打开哪一个面板。

支持多个类型：

- **Boolean**：设置 `active` 为 `false` 将折叠所有的面板。这要求 `collapsible` 选项必须为 `true` 。
- **Integer**：激活打开的面板索引，以零为基础。负值则表示从最后一个面板后退选择面板。

代码实例：

初始化带有指定 `active` 选项的 accordion：

```
$( ".selector" ).accordion({ active: 2 });
```

在初始化后，获取或设置 `active` 选项：

```
// getter
var active = $( ".selector" ).accordion( "option", "active" );

// setter
$( ".selector" ).accordion( "option", "active", 2 );
```

默认值：0

animate

类型：Boolean 或 Number 或 String 或 Object

描述：是否使用动画改变面板，且如何使用动画改变面板。

支持多个类型：

- **Boolean**： `false` 值将禁用动画。
- **Number**：easing 默认的持续时间，以毫秒计。
- **String**：默认的持续时间要使用的 easing 名称。
- **Object**： `easing` 和 `duration` 属性的动画设置。

- 上面任意的选项都可以包含 `down` 属性。
- 当被激活的面板有一个比当前激活面板较低的指数时，发生 "Down" 动画。

代码实例：

初始化带有指定 `animate` 选项的 accordion：

```
$( ".selector" ).accordion({ animate: "bounceslide" });
```

在初始化后，获取或设置 `animate` 选项：

```
// getter
var animate = $( ".selector" ).accordion( "option", "animate" );

// setter
$( ".selector" ).accordion( "option", "animate", "bounceslide" );
```

默认值：{}

collapsible

类型：Boolean

描述：所有部分是否都可以马上关闭。允许折叠激活的部分。

代码实例：

初始化带有指定 `collapsible` 选项的 accordion：

```
$( ".selector" ).accordion({ collapsible: true });
```

在初始化后，获取或设置 `collapsible` 选项：

```
// getter
var collapsible = $( ".selector" ).accordion( "option", "collapsibl

// setter
$( ".selector" ).accordion( "option", "collapsible", true );
```

默认值：false

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 accordion。

代码实例：

初始化带有指定 `disabled` 选项的 accordion：

```
$( ".selector" ).accordion({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).accordion( "option", "disabled" );

// setter
$( ".selector" ).accordion( "option", "disabled", true );
```

默认值：false

event

类型：String

描述：accordion 头部会作出反应的事件，用以激活相关的面板。可以指定多个事件，用空格间隔。

代码实例：

初始化带有指定 `event` 选项的 accordion：

```
$( ".selector" ).accordion({ event: "mouseover" });
```

在初始化后，获取或设置 `event` 选项：

```
// getter
var event = $( ".selector" ).accordion( "option", "event" );

// setter
$( ".selector" ).accordion( "option", "event", "mouseover" );
```

默认值："click"

header

类型：Selector

描述：标题元素的选择器，通过主要 accordion 元素上的 .find() 进行应用。内容面板必须是紧跟在与其相关的标题后的同级元素。

代码实例：

初始化带有指定 `header` 选项的 accordion：

```
$( ".selector" ).accordion({ header: "h3" });
```

在初始化后，获取或设置 `header` 选项：

```
// getter
var header = $( ".selector" ).accordion( "option", "header" );

// setter
$( ".selector" ).accordion( "option", "header", "h3" );
```

默认值："> li > :first-child,> :not(li):even"

heightStyle

类型：String

描述：控制 accordion 和每个面板的高度。可能的值：

- `"auto"` ：所有的面板将会被设置为最高的面板的高度。
- `"fill"` ：基于 accordion 的父元素的高度，扩展到可用的高度。
- `"content"` ：每个面板的高度取决于它的内容。

代码实例：

初始化带有指定 `heightStyle` 选项的 accordion：

```
$( ".selector" ).accordion({ heightStyle: "fill" });
```

在初始化后，获取或设置 `heightStyle` 选项：

```
// getter
var heightStyle = $( ".selector" ).accordion( "option", "heightSty

// setter
$( ".selector" ).accordion( "option", "heightStyle", "fill" );
```

默认值："auto"

icons

类型：Object

描述：标题要使用的图标，与 jQuery UI CSS 框架提供的图标（Icons） 匹配。设置为 `false` 则不显示图标。

- header （string，默认值："ui-icon-triangle-1-e"）
- activeHeader （string，默认值："ui-icon-triangle-1-s"）

代码实例：

初始化带有指定 `icons` 选项的 accordion：

```
$( ".selector" ).accordion({ icons: { "header": "ui-icon-plus", "ac
```

在初始化后，获取或设置 `icons` 选项：

```
// getter
var icons = $( ".selector" ).accordion( "option", "icons" );

// setter
$( ".selector" ).accordion( "option", "icons", { "header": "ui-icon
```

默认值：{ "header": "ui-icon-triangle-1-e", "activeHeader": "ui-icon-triangle-1-s" }

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 accordion 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).accordion( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 accordion。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).accordion( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 accordion。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).accordion( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).accordion( "option", "disabled" );
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 accordion 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).accordion( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 accordion 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).accordion( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 accordion 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).accordion( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：处理任何在 DOM 中直接添加或移除的标题和面板，并重新计算 accordion 的高度。结果取决于内容和 `[heightStyle](#option-heightStyle )` 选项。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).accordion( "refresh" );
```

widget()

类型：jQuery

描述：返回一个包含 accordion 的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).accordion( "widget" );
```

事件

activate(event, ui)

类型：accordionactivate

描述：面板被激活后触发（在动画完成之后）。如果 accordion 之前是折叠的，则 `ui.oldHeader` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newHeader` 和 `ui.newPanel` 将是空的 jQuery 对象。

注意：由于 **`activate`** 事件只有在面板激活时才能触发，当创建 **accordion** 部件时，最初的面板不会触发该事件。如果您需要一个用于部件创建的钩，请使用 **`create`** 事件。

- **event** 类型：Event
- **ui** 类型：Object
  - **newHeader** 类型：jQuery 描述：刚被激活的标题。
  - **oldHeader** 类型：jQuery 描述：刚被取消激活的标题。
  - **newPanel** 类型：jQuery 描述：刚被激活的面板。
  - **oldPanel** 类型：jQuery 描述：刚被取消激活的面板。

代码实例：

初始化带有指定 activate 回调的 accordion：

```
$( ".selector" ).accordion({
  activate: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordionactivate 事件：

```
$( ".selector" ).on( "accordionactivate", function( event, ui ) {}
```

beforeActivate(event, ui)

类型：accordionbeforeactivate

描述：面板被激活前直接触发。可以取消以防止面板被激活。如果 accordion 当前是折叠的，则 `ui.oldHeader` 和 `ui.oldPanel` 将是空的 jQuery 对象。如果 accordion 正在折叠，则 `ui.newHeader` 和 `ui.newPanel` 将是空的 jQuery 对象。

- **event** 类型：Event
- **ui** 类型：Object
  - **newHeader** 类型：jQuery 描述：将被激活的标题。
  - **oldHeader** 类型：jQuery 描述：将被取消激活的标题。

- **newPanel** 类型：jQuery 描述：将被激活的面板。
- **oldPanel** 类型：jQuery 描述：将被取消激活的面板。

代码实例：

初始化带有指定 beforeActivate 回调的 accordion：

```
$( ".selector" ).accordion({
  beforeActivate: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordionbeforeactivate 事件：

```
$( ".selector" ).on( "accordionbeforeactivate", function( event, ui
```

create(event, ui)

类型：accordioncreate

描述：当创建 accordion 时触发。如果 accordion 是折叠的，`ui.header` 和 `ui.panel` 将是空的 jQuery 对象。

- **event** 类型：Event
- **ui** 类型：Object
  - **header** 类型：jQuery 描述：激活的标题。
  - **panel** 类型：jQuery 描述：激活的面板。

代码实例：

初始化带有指定 create 回调的 accordion：

```
$( ".selector" ).accordion({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 accordioncreate 事件：

```
$( ".selector" ).on( "accordioncreate", function( event, ui ) {} );
```

实例

一个简单的 jQuery UI 折叠面板（Accordion）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>折叠面板部件（Accordion Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="accordion">
  <h3>部分 1</h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget.
    Integer ut neque. Vivamus nisi metus, molestie vel, gravida in,
    condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros.
    Nam mi. Proin viverra leo ut odio.</p>
  </div>
  <h3>部分 2</h3>
  <div>
    <p>Sed non urna. Phasellus eu ligula. Vestibulum sit amet purus
    Vivamus hendrerit, dolor aliquet laoreet, mauris turpis velit,
    faucibus interdum tellus libero ac justo.</p>
  </div>
  <h3>部分 3</h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus.
    Quisque lobortis.Phasellus pellentesque purus in massa.</p>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
</div>

<script>
$( "#accordion" ).accordion();
</script>

</body>
</html>
```

jQuery UI API - 自动完成部件（Autocomplete Widget）

所属类别

小部件（Widgets）

用法

描述：自动完成功能根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。

版本新增：1.8

任何可以接收输入的字段都可以转换为 Autocomplete，即，`<input>` 元素，`<textarea>` 元素及带有 `contenteditable` 属性的元素。

通过给 Autocomplete 字段焦点或者在其中输入字符，插件开始搜索匹配的条目并显示供选择的值的列表。通过输入更多的字符，用户可以过滤列表以获得更好的匹配。

该部件可用于选择先前选定的值，比如输入文章标签或者输入从地址簿中输入地址邮件地址。Autocomplete 也可以用来填充相关的信息，比如输入一个城市的名称来获得该城市的邮政编码。

您可以从本地源或者远程源获取数据：本地源适用于小数据集，例如，带有 50 个条目的地址簿；远程源适用于大数据集，比如，带有数百个或者成千上万个条目的数据库。如需了解更多有关自定义数据源的信息，请查看 `source` 选项的文档。

键盘交互

当菜单打开时，下面的键盘命令可用：

- UP - 移动焦点到上一个项目。如果在第一个项目上，则移动焦点到输入（input）。如果在输入（input）上，则移动焦点到最后一个项目。
- DOWN - 移动焦点到下一个项目。如果在最后一个项目上，则移动焦点到输入（input）。如果在输入（input）上，则移动焦点到第一个项目。
- ESCAPE - 关闭菜单。
- ENTER - 选择当前获得焦点的项目，并关闭菜单。
- TAB - 选择当前获得焦点的项目，关闭菜单，并移动焦点到下一个可聚焦（focusable）的元素。
- PAGE UP/DOWN - 滚动一屏的项目（基于菜单的高度）。

当菜单关闭时，下面的键盘命令可用：

- UP/DOWN - 如果满足 `minLength`，则打开菜单。

主题化

自动完成部件（Autocomplete Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用自动完成部件指定的样式，则可以使用下面的 CSS class 名称：

- `ui-autocomplete`：用于显示匹配用户的 菜单（menu）
- `ui-autocomplete-input`：自动完成部件（Autocomplete Widget）实例化的 input 元素。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 菜单部件（Menu Widget）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。

快速导航

选项	
appendToautoFocusdelaydisabledminLengthpositionsource	closedestroydisab

appendTo

类型：Selector

描述：菜单应该被附加到哪一个元素。当该值为 `null` 时，输入域的父元素将检查 `ui-front` class。如果找到带有 `ui-front` class 的元素，菜单将被附加到该元素。如果未找到带有 `ui-front` class 的元素，不管值为多少，菜单将被附加到 body。

注意：当建议菜单打开时，**`appendTo`** 选项不应改变。

代码实例：

初始化带有指定 `appendTo` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ appendTo: "#someElem" });
```

在初始化后，获取或设置 `appendTo` 选项：

```
// getter
var appendTo = $( ".selector" ).autocomplete( "option", "appendTo"

// setter
$( ".selector" ).autocomplete( "option", "appendTo", "#someElem" );
```

默认值：null

autoFocus

类型：Boolean

描述：如果设置为 `true`，当菜单显示时，第一个条目将自动获得焦点。

代码实例：

初始化带有指定 `autoFocus` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ autoFocus: true });
```

在初始化后，获取或设置 `autoFocus` 选项：

```
// getter
var autoFocus = $( ".selector" ).autocomplete( "option", "autoFocus

// setter
$( ".selector" ).autocomplete( "option", "autoFocus", true );
```

默认值：false

delay

类型：Integer

描述：按键和执行搜索之间的延迟，以毫秒计。对于本地数据，采用零延迟是有意义的（更具响应性），但对于远程数据会产生大量的负荷，同时降低了响应性。

代码实例：

初始化带有指定 `delay` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ delay: 500 });
```

在初始化后，获取或设置 `delay` 选项：

```
// getter
var delay = $( ".selector" ).autocomplete( "option", "delay" );

// setter
$( ".selector" ).autocomplete( "option", "delay", 500 );
```

默认值：300

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 autocomplete。

代码实例：

初始化带有指定 `disabled` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).autocomplete( "option", "disabled"

// setter
$( ".selector" ).autocomplete( "option", "disabled", true );
```

默认值：false

minLength

类型：Integer

描述：执行搜索前用户必须输入的最小字符数。对于仅带有几项条目的本地数据，通常设置为零，但是当单个字符搜索会匹配几千项条目时，设置个高数值是很有必要的。

代码实例：

初始化带有指定 `minLength` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ minLength: 0 });
```

在初始化后，获取或设置 `minLength` 选项：

```
// getter
var minLength = $( ".selector" ).autocomplete( "option", "minLength

// setter
$( ".selector" ).autocomplete( "option", "minLength", 0 );
```

默认值：1

position

类型：Object

描述：标识建议菜单的位置与相关的 input 元素有关系。 `of` 选项默认为 input 元素，但是您可以指定另一个定位元素。如需了解各种选项的更多细节，请查看 jQuery UI 定位（Position）。

代码实例：

初始化带有指定 `position` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ position: { my : "right top", at: '
```

在初始化后，获取或设置 `position` 选项：

```
// getter
var position = $( ".selector" ).autocomplete( "option", "position"

// setter
$( ".selector" ).autocomplete( "option", "position", { my : "right
```

默认值：{ my: "left top", at: "left bottom", collision: "none" }

source

类型：Array 或 String 或 Function(Object request, Function response(Object data))

描述：定义要使用的数据，必须指定。

独立于您要使用的变量，标签总是被视为文本。如果您想要标签被视为 html，您可以使用 Scott González' html 扩展。演示侧重于 `source` 选项的不同变量 - 您可以查找其中匹配您的使用情况的那个，并查看相关代码。

支持多个类型：

- **Array**：可用于本地数据的一个数组。支持两种格式：
  - 字符串数组：`[ "Choice1", "Choice2" ]`
  - 带有 `label` 和 `value` 属性的对象数组：`[ { label: "Choice1", value: "value1" }, ... ]` label 属性显示在建议菜单中。当用户选择一个条目时，value 将被插入到 input 元素中。如果只是指定了一个属性，则该属性将被视为 label 和 value，例如，如果您只提供了 `value` 属性，value 也会被视为标签。
- **String**：当使用一个字符串，Autocomplete 插件希望该字符串指向一个能返回 JSON 数据的 URL 资源。它可以是在相同的主机上，也可以是在不同的主机上（必须提供 JSONP）。Autocomplete 插件不过滤结果，而是通过一个 `term` 字段添加了一个查询字符串，用于服务器端脚本过滤结果。例如，如果 `source` 选项设置为 `"http://example.com"`，且用户键入了 `foo`，GET 请求则为 `http://example.com?term=foo`。数据本身的格式可以与前面描述的本地数据的格式相同。
- **Function**：第三个变量，一个回调函数，提供最大的灵活性，可用于连接任何数据源到 Autocomplete。回调函数接受两个参数：
  - 一个 `request` 对象，带有一个 `term` 属性，表示当前文本输入中的值。例如，如果用户在 city 字段输入 `"new yo"`，则 Autocomplete term 等同于 `"new yo"`。
  - 一个 `response` 回调函数，提供单个参数：建议给用户的数据。该数据应基于被提供的 term 进行过滤，且可以是上面描述的本地数据的任何格式。用于在请求期间提供自定义 source 回调来处理错误。即使遇到错误，您也必须调用 `response` 回调函数。这就确保了小部件总是正确的状态。

  当过滤本地数据时，您可以使用内置的 `$.ui.autocomplete.escapeRegex` 函数。它会接受一个字符串参数，转义所有的正则表达式字符，让结果安全地传递到 `new RegExp()`。

代码实例：

初始化带有指定 `source` 选项的 autocomplete：

```
$( ".selector" ).autocomplete({ source: [ "c++", "java", "php", "co
```

在初始化后，获取或设置 `source` 选项：

```
// getter
var source = $( ".selector" ).autocomplete( "option", "source" );

// setter
$( ".selector" ).autocomplete( "option", "source", [ "c++", "java",
```

默认值：none; must be specified

方法

close()

类型：jQuery (plugin only)

描述：关闭 Autocomplete 菜单。当与 `[search](#method-search)` 方法结合使用时，可用于关闭打开的菜单。

- 该方法不接受任何参数。

代码实例：

调用 close 方法：

```
$( ".selector" ).autocomplete( "close" );
```

destroy()

类型：jQuery (plugin only)

描述：完全移除 autocomplete 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).autocomplete( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 autocomplete。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).autocomplete( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 autocomplete。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).autocomplete( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).autocomplete( "option", "disablec
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 autocomplete 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).autocomplete( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 autocomplete 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).autocomplete( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 autocomplete 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).autocomplete( "option", { disabled: true } );
```

search([value])

类型：jQuery (plugin only)

描述：触发 `search` 事件，如果该事件未被取消则调用数据源。当被点击时，可被类似选择框按钮用来打开建议。当不带参数调用该方法时，则使用当前输入的值。可带一个空字符串和 `minLength: 0` 进行调用，来显示所有的条目。

- **value** 类型：String

代码实例：

调用 search 方法：

```
$( ".selector" ).autocomplete( "search", "" );
```

widget()

类型：jQuery

描述：返回一个包含菜单元素的 `jQuery` 对象。虽然菜单项不断地被创建和销毁。菜单元素本身会在初始化时创建，并不断的重复使用。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
$( ".selector" ).autocomplete( "widget" );
```

自动完成部件（Autocomplete Widget）通过 部件库（Widget Factory） 创建的，且可被扩展。当对部件进行扩展时，您可以重载或者添加扩展部件的行为。下面提供的方法作为扩展点，与上面列出的 插件方法 具有相同的 API 稳定性。如需了解更多有关小部件扩展的知识，请查看 通过部件库（Widget Factory）扩展小部件.

_renderItem(ul, item)

类型：jQuery

描述：Method that controls the creation of each option in the widget's menu. The method must create a new <li> element, append it to the menu, and return it.

Note: At this time the <ul> element created must contain an <a> element for compatibility with the menu widget. See the example below.

- **ul** 类型：jQuery 描述：新创建的 <li> 元素必须追加到的 <ul> 元素。
- **item** 类型：Object
    - **label** 类型：String 描述：条目显示的字符串。
    - **item** 类型：String 描述：当条目被选中时插入到输入框中的值。

代码实例：

添加条目的值作为 <li> 上的 data 属性。

```
_renderItem: function( ul, item ) {
  return $( "&lt;li&gt;" )
    .attr( "data-value", item.value )
    .append( $( "&lt;a&gt;" ).text( item.label ) )
    .appendTo( ul );
}
```

_renderMenu(ul, items)

类型：jQuery (plugin only)

描述：该方法负责在菜单显示前调整菜单尺寸。菜单元素可通过 `this.menu.element` 使用。

- **ul** 类型：jQuery 描述：一个要作为小部件的菜单使用的空的 `<ul>` 元素。
- **items** 类型：Array 描述：一个数组，数组元素为匹配用户输入的条目。每个条目是一个带有 `label` 和 `value` 属性的对象。

代码实例：

添加一个 CSS class 名称到旧的菜单项。

```
_renderMenu: function( ul, items ) {
  var that = this;
  $.each( items, function( index, item ) {
    that._renderItemData( ul, item );
  });
  $( ul ).find( "li:odd" ).addClass( "odd" );
}
```

_resizeMenu()

类型：jQuery (plugin only)

描述：该方法负责在菜单显示前调整菜单尺寸。菜单元素可通过 `this.menu.element` 使用。

- 该方法不接受任何参数。

代码实例：

菜单总是显示为 500 像素宽。

```
_resizeMenu: function() {
  this.menu.element.outerWidth( 500 );
}
```

事件

change(event, ui)

类型：autocompletechange

描述：如果输入域的值改变则触发该事件。

- **event** 类型：Event
- **ui** 类型：Object
    - **item** 类型：Object 描述：从菜单中选择的条目，否则属性为 `null` 。

代码实例：

初始化带有指定 change 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  change: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletechange 事件：

```
$( ".selector" ).on( "autocompletechange", function( event, ui ) {}
```

close(event, ui)

类型：autocompleteclose

描述：当菜单隐藏时触发。不是每一个 `close` 事件都伴随着 `change` 事件。

- **event** 类型：Event
- **ui** 类型：Object

注意： `ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 close 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  close: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteclose 事件：

```
$( ".selector" ).on( "autocompleteclose", function( event, ui ) {}
```

create(event, ui)

类型：autocompletecreate

描述：当创建 autocomplete 时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletecreate 事件：

```
$( ".selector" ).on( "autocompletecreate", function( event, ui ) {
```

focus(event, ui)

类型：autocompletefocus

描述：当焦点移动到一个条目上（未选择）时触发。默认的动作是把文本域中的值替换为获得焦点的条目的值，即使该事件是通过键盘交互触发的。取消该事件会阻止值被更新，但不会阻止菜单项获得焦点。

- **event** 类型：Event
- **ui** 类型：Object
    - **item** 类型：Object 描述：获得焦点的条目。

代码实例：

初始化带有指定 focus 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  focus: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletefocus 事件：

```
$( ".selector" ).on( "autocompletefocus", function( event, ui ) {}
```

open(event, ui)

类型：autocompleteopen

描述：当打开建议菜单或者更新建议菜单时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 open 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  open: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteopen 事件：

```
$( ".selector" ).on( "autocompleteopen", function( event, ui ) {} 
```

response(event, ui)

类型：autocompleteresponse

描述：在搜索完成后菜单显示前触发。用于建议数据的本地操作，其中自定义的 `source` 选项回调不是必需的。该事件总是在搜索完成时触发，如果搜索无结果或者禁用了 Autocomplete，导致菜单未显示，该事件一样会被触发。

- **event** 类型：Event
- **ui** 类型：Object
  - **content** 类型：Array 描述：包含响应数据，且可被修改来改变显示结果。该数据已经标准化，所以如果您要修改数据，请确保每个条目都包含 `value` 和 `label` 属性。

代码实例：

初始化带有指定 response 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  response: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteresponse 事件：

```
$( ".selector" ).on( "autocompleteresponse", function( event, ui )
```

search(event, ui)

类型：autocompletesearch

描述：在搜索执行前满足 `minLength` 和 `delay` 后触发。如果取消该事件，则不会提交请求，也不会提供建议条目。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 search 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  search: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompletesearch 事件：

```
$( ".selector" ).on( "autocompletesearch", function( event, ui ) {
```

select(event, ui)

类型：autocompleteselect

描述：当从菜单中选择条目时触发。默认的动作是把文本域中的值替换为被选中的条目的值。取消该事件会阻止值被更新，但不会阻止菜单关闭。

- **event** 类型：Event
- **ui** 类型：Object
  - **item** 类型：Object 描述：一个带有被选项的 `label` 和 `value` 属性的对象。

代码实例：

初始化带有指定 select 回调的 autocomplete：

```
$( ".selector" ).autocomplete({
  select: function( event, ui ) {}
});
```

绑定一个事件监听器到 autocompleteselect 事件：

```
$( ".selector" ).on( "autocompleteselect", function( event, ui ) {}
```

实例

实例 **1**：

一个简单的 jQuery UI 自动完成（Autocomplete）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>自动完成部件（Autocomplete Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<label for="autocomplete">选择一个编程语言：</label>
<input id="autocomplete">

<script>
$( "#autocomplete" ).autocomplete({
  source: [ "c++", "java", "php", "coldfusion", "javascript", "asp'
});
</script>

</body>
</html>
```

实例 **2**：

使用自定义源回调来匹配条件的开始。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>自动完成部件（Autocomplete Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<label for="autocomplete">选择一个编程语言：</label>
<input id="autocomplete">

<script>
var tags = [ "c++", "java", "php", "coldfusion", "javascript", "asp
$( "#autocomplete" ).autocomplete({
  source: function( request, response ) {
          var matcher = new RegExp( "^" + $.ui.autocomplete.escapeR
          response( $.grep( tags, function( item ){
              return matcher.test( item );
          }) );
      }
});
</script>

</body>
</html>
```

jQuery UI API - 按钮部件（Button Widget）

所属类别

小部件（Widgets）

用法

描述：可主题化的按钮和按钮集合。

版本新增：1.8

按钮部件（Button Widget）加强了标准表单元素的功能，比如按钮（button）、输入（input）、锚（anchor），用适当的悬停（hover）和激活（active）样式来主题化按钮。

除了基本的按钮，单选按钮和复选框（input 类型为 radio 和 checkbox）也可以转换为按钮。相关的标签（label）设计成按钮的样式，点击时更新底层的输入。为了能正常工作，需要给 input 一个 `id` 属性，并指向标签（label）的 `for` 属性。不要把 input 放在标签（label）内，否则会引起可访问性问题。

为了分组单选按钮，Button 也提供了一个额外的小部件，名为 Buttonset。Buttonset 通过选择一个容器元素（包含单选按钮）并调用 `.buttonset()` 来使用。Buttonset 也提供了可视化分组，因此当有一组按钮时都可考虑使用它。它会选择所有的后代，并对它们应用 `.button()`。您可以启用和禁用一个按钮集，这将会启用和禁用所有包含的按钮。销毁按钮集会调用每个按钮的 `destroy` 方法。对于分组的单选按钮和复选框按钮，推荐使用带有 `legend` 的 `fieldset` 来提供一个可访问的分组标签。

当使用一个类型为 button、submit 或 reset 的 input 时，仅限于支持纯文本无图标标签。

主题化

按钮部件（Button Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用按钮指定的样式，则可以使用下面的 CSS class 名称：

- `ui-button`：表示按钮的 DOM 元素。该元素会根据 text 和 icons 选项添加下列 class 之一：`ui-button-text-only`、`ui-button-icon-only`、`ui-button-icon`、`ui-button-text-icons`。
  - `ui-button-icon-primary`：用于显示按钮主要图标的元素。只有当主要图标在 icons 选项中提供时才呈现。
  - `ui-button-text`：在按钮的文本内容周围的容器。

    - `ui-button-icon-secondary`：用于显示按钮的次要图标。只有当次要图标在 icons 选项中提供时才呈现。
- `ui-buttonset`：Buttonset 的外层容器。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

快速导航

选项	方法	事件
disablediconslabeltext	destroydisableenableoptionrefreshwidget	create

disabled

类型：Boolean

描述：如果设置为 `true`，则禁用该 button。

代码实例：

初始化带有指定 `disabled` 选项的 button：

```
$( ".selector" ).button({ disabled: true });
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var disabled = $( ".selector" ).button( "option", "disabled" );

// setter
$( ".selector" ).button( "option", "disabled", true );
```

默认值：false

icons

类型：Object

描述：要显示的图标，包括带有文本的图标和不带有文本的图标（查看 `text` 选项）。默认情况下，主图标显示在标签文本的左边，副图标显示在右边。显示位置可通过 CSS 进行控制。

`primary` 和 `secondary` 属性值必须是 图标 class 名称，例如，`"ui-icon-gear"` 。如果只使用一个图标，则 `icons: { primary: "ui-icon-locked" }` 。如果使用两个图标，则 `icons: { primary: "ui-icon-gear", secondary: "ui-icon-triangle-1-s"` 。

代码实例：

初始化带有指定 `icons` 选项的 button：

```
$( ".selector" ).button({ icons: { primary: "ui-icon-gear", seconda
```

在初始化后，获取或设置 `disabled` 选项：

```
// getter
var icons = $( ".selector" ).button( "option", "icons" );

// setter
$( ".selector" ).button( "option", "icons", { primary: "ui-icon-gea
```

默认值：{ primary: null, secondary: null }

label

类型：String

描述：要显示在按钮中的文本。当未指定时（ `null` ），则使用元素的 HTML 内容，或者如果元素是一个 submit 或 reset 类型的 input 元素，则使用它的 `value` 属性，或者如果元素是一个 radio 或 checkbox 类型的 input 元素，则使用相关的 label 元素的 HTML 内容。

代码实例：

初始化带有指定 `label` 选项的 button：

```
$( ".selector" ).button({ label: "custom label" });
```

在初始化后，获取或设置 `label` 选项：

```
// getter
var label = $( ".selector" ).button( "option", "label" );

// setter
$( ".selector" ).button( "option", "label", "custom label" );
```

默认值：null

text

类型：Boolean

描述：是否显示标签。当设置为 `false` 时，不显示文本，但是此时必须启用 `icons` 选项，否则 `text` 选项将被忽略。

代码实例：

初始化带有指定 `text` 选项的 button：

```
$( ".selector" ).button({ text: false });
```

在初始化后，获取或设置 `text` 选项：

```
// getter
var text = $( ".selector" ).button( "option", "text" );

// setter
$( ".selector" ).button( "option", "text", false );
```

默认值：true

方法

destroy()

类型：jQuery (plugin only)

描述：完全移除 button 功能。这会把元素返回到它的预初始化状态。

- 该方法不接受任何参数。

代码实例：

调用 destroy 方法：

```
$( ".selector" ).button( "destroy" );
```

disable()

类型：jQuery (plugin only)

描述：禁用 button。

- 该方法不接受任何参数。

代码实例：

调用 disable 方法：

```
$( ".selector" ).button( "disable" );
```

enable()

类型：jQuery (plugin only)

描述：启用 button。

- 该方法不接受任何参数。

代码实例：

调用 enable 方法：

```
$( ".selector" ).button( "enable" );
```

option(optionName)

类型：Object

描述：获取当前与指定的 `optionName` 关联的值。

- **optionName** 类型：String 描述：要获取的选项的名称。

代码实例：

调用该方法：

```
var isDisabled = $( ".selector" ).button( "option", "disabled" );
```

option()

类型：PlainObject

描述：获取一个包含键/值对的对象，键/值对表示当前 button 选项哈希。

- 该方法不接受任何参数。

代码实例：

调用该方法：

```
var options = $( ".selector" ).button( "option" );
```

option(optionName, value)

类型：jQuery (plugin only)

描述：设置与指定的 `optionName` 关联的 button 选项的值。

- **optionName** 类型：String 描述：要设置的选项的名称。
- **value** 类型：Object 描述：要为选项设置的值。

代码实例：

调用该方法：

```
$( ".selector" ).button( "option", "disabled", true );
```

option(options)

类型：jQuery (plugin only)

描述：为 button 设置一个或多个选项。

- **options** 类型：Object 描述：要设置的 option-value 对。

代码实例：

调用该方法：

```
$( ".selector" ).button( "option", { disabled: true } );
```

refresh()

类型：jQuery (plugin only)

描述：刷新按钮的视觉状态。用于在以编程方式改变原生元素的选中状态或禁用状态后更新按钮状态。

- 该方法不接受任何参数。

代码实例：

调用 refresh 方法：

```
$( ".selector" ).button( "refresh" );
```

widget()

类型：jQuery

描述：返回一个包含 button 的 `jQuery` 对象。

- 该方法不接受任何参数。

代码实例：

调用 widget 方法：

```
var widget = $( ".selector" ).button( "widget" );
```

事件

create(event, ui)

类型：buttoncreate

描述：当创建按钮 button 时触发。

- **event** 类型：Event
- **ui** 类型：Object

注意：`ui` 对象是空的，这里包含它是为了与其他事件保持一致性。

代码实例：

初始化带有指定 create 回调的 button：

```
$( ".selector" ).button({
  create: function( event, ui ) {}
});
```

绑定一个事件监听器到 buttoncreate 事件：

```
$( ".selector" ).on( "buttoncreate", function( event, ui ) {} );
```

实例

实例 **1**：

一个简单的 jQuery UI 按钮（Button）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>按钮部件（Button Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button>按钮标签</button>

<script>
$( "button" ).button();
</script>

</body>
</html>
```

实例 **2**：

一个简单的 jQuery UI 按钮集（Buttonset）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>按钮部件（Button Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<form>
  <fieldset>
    <legend>最喜欢的 jQuery 项目</legend>
    <div id="radio">
      <input type="radio" id="sizzle" name="project">
      <label for="sizzle">Sizzle</label>

      <input type="radio" id="qunit" name="project" checked="checke
      <label for="qunit">QUnit</label>

      <input type="radio" id="color" name="project">
      <label for="color">Color</label>
    </div>
  </fieldset>
</form>

<script>
$( "#radio" ).buttonset();
</script>

</body>
</html>
```

jQuery UI API - 日期选择器部件（Datepicker Widget）

所属类别

小部件（Widgets）

用法

描述：从弹出框或内联日历选择一个日期。

版本新增：1.0

jQuery UI 日期选择器（Datepicker）是向页面添加日期选择功能的高度可配置插件。您可以自定义日期格式和语言，限制可选择的日期范围，添加按钮和其他导航选项。

默认情况下，当相关的文本域获得焦点时，在一个小的覆盖层打开日期选择器。对于一个内联的日历，只需简单地将日期选择器附加到 div 或者 span 上。

键盘交互

当日期选择器打开时，下面的键盘命令可用：

- PAGE UP：移到上一个月。
- PAGE DOWN：移到下一个月。
- CTRL+PAGE UP：移到上一年。
- CTRL+PAGE DOWN：移到下一年。
- CTRL+HOME：移到当前月份。如果日期选择器是关闭的则打开。
- CTRL+LEFT：移到上一天。
- CTRL+RIGHT：移到下一天。
- CTRL+UP：移到上一周。
- CTRL+DOWN：移到下一周。
- ENTER：选择聚焦的日期。
- CTRL+END：关闭日期选择器，并清除日期。
- ESCAPE：关闭日期选择器，不做任何选择。

实用功能

$.datepicker.setDefaults(settings)

为所有的日期选择器改变默认设置。

使用 `option()` 方法来改变个别实例的设置。

代码实例：

设置所有的日期选择器在获得焦点时或点击图标时打开。

```
$.datepicker.setDefaults({
  showOn: "both",
  buttonImageOnly: true,
  buttonImage: "calendar.gif",
  buttonText: "Calendar"
});
```

设置所有的日期选择器都有法语文本。

```
$.datepicker.setDefaults( $.datepicker.regional[ "fr" ] );
```

$.datepicker.formatDate(format, date, settings)

格式化日期为一个带有指定格式的字符串值。

格式可以是下列组合：

- d - 一月中的第几天（没有前导零）
- dd - 一月中的第几天（两位数）
- o - 一年中的第几天（没有前导零）
- oo - 一年中的第几天（三位数）
- D - 天的短名称
- DD - 天的长名称
- m - 一年中的第几月（没有前导零）
- mm - 一年中的第几月（两位数）
- M - 月的短名称
- MM - 月的长名称
- y - 年（两位数）
- yy - 年（四位数）
- @ - Unix 时间戳（ms since 01/01/1970）
- ! - Windows 钟表（100ns since 01/01/0001）
- '...' - 文本
- '' - 单引号
- 其他 - 文本

也有一些 `$.datepicker` 预定义的标准日期格式：

- ATOM - 'yy-mm-dd' （与 RFC 3339/ISO 8601 相同）
- COOKIE - 'D, dd M yy'
- ISO_8601 - 'yy-mm-dd'
- RFC_822 - 'D, d M y' （参照 RFC 822）
- RFC_850 - 'DD, dd-M-y' （参照 RFC 850）

- RFC_1036 - 'D, d M y' （参照 RFC 1036）
- RFC_1123 - 'D, d M yy' （参照 RFC 1123）
- RFC_2822 - 'D, d M yy' （参照 RFC 2822）
- RSS - 'D, d M y' （与 RFC 822 相同）
- TICKS - '!'
- TIMESTAMP - '@'
- W3C - 'yy-mm-dd' （与 ISO 8601 相同）

代码实例：

以 ISO 格式显示日期。产生 "2007-01-26"。

```
$.datepicker.formatDate( "yy-mm-dd", new Date( 2007, 1 - 1, 26 ) );
```

以扩展法语格式显示日期。产生 "Samedi, Juillet 14, 2007"。

```
$.datepicker.formatDate( "DD, MM d, yy", new Date( 2007, 7 - 1, 14
  dayNamesShort: $.datepicker.regional[ "fr" ].dayNamesShort,
  dayNames: $.datepicker.regional[ "fr" ].dayNames,
  monthNamesShort: $.datepicker.regional[ "fr" ].monthNamesShort,
  monthNames: $.datepicker.regional[ "fr" ].monthNames
});
```

$.datepicker.parseDate(format, value, settings)

从一个指定格式的字符串值中提取日期。

格式可以是下列组合：

- d - 一月中的第几天（没有前导零）
- dd - 一月中的第几天（两位数）
- o - 一年中的第几天（没有前导零）
- oo - 一年中的第几天（三位数）
- D - 星期几的短名称
- DD - 星期几的长名称
- m - 一年中的第几月（没有前导零）
- mm - 一年中的第几月（两位数）
- M - 月的短名称
- MM - 月的长名称
- y - 年（两位数）
- yy - 年（四位数）
- @ - Unix 时间戳（ms since 01/01/1970）
- ! - Windows 钟表（100ns since 01/01/0001）
- '...' - 文本
- '' - 单引号

- 其他 - 文本

一些可能被抛出的异常：

- 'Invalid arguments' - 如果格式或值为空则抛出此异常。
- 'Missing number at position nn' - 如果格式显示一个未找到的数值则抛出此异常。
- 'Unknown name at position nn' - 如果格式显示一个未找到的星期几名称或月份名称则抛出此异常。
- 'Unexpected literal at position nn' - 如果格式显示一个未找到的文本值则抛出此异常。
- 'Invalid date' - 如果日期无效则抛出此异常，比如 '31/02/2007'。

代码实例：

提取一个 ISO 格式的日期。

```
$.datepicker.parseDate( "yy-mm-dd", "2007-01-26" );
```

提取一个扩展法语格式的日期。

```
$.datepicker.parseDate( "DD, MM d, yy", "Samedi, Juillet 14, 2007",
  shortYearCuroff: 20,
  dayNamesShort: $.datepicker.regional[ "fr" ].dayNamesShort,
  dayNames: $.datepicker.regional[ "fr" ].dayNames,
  monthNamesShort: $.datepicker.regional[ "fr" ].monthNamesShort,
  monthNames: $.datepicker.regional[ "fr" ].monthNames
});
```

$.datepicker.iso8601Week(date)

确定一个给定的日期在一年中的第几周：1 到 53。

该函数使用 ISO 8601 定义一周：一周从星期一开始，每一年的第一周包含 1 月 4 日。这意味着上一年至多有三天可能包含在当年的第一周中，当年至多有三天可能包含在上一年的最后一周中。

该函数是 `calculateWeek` 选项的默认实现。

代码实例：

查找日期在一年中的第几周。

```
$.datepicker.iso8601Week( new Date( 2007, 1 - 1, 26 ) );
```

$.datepicker.noWeekends

设置如 beforeShowDay 函数，防止选择周末。

我们可以在 `beforeShowDay` 选项中提供 `noWeekends()` 函数，用来计算所有工作日，提供一个 `true / false` 值的数组，用来指示日期是否可选择。

代码实例：

设置 DatePicker，让周末不可选。

```
$( "#datepicker" ).datepicker({
  beforeShowDay: $.datepicker.noWeekends
});
```

局限

日期选择器提供了迎合不同的语言和日期格式本地化内容的支持。每个本地化包含在名称后追加语言代码的文件中，例如法语为 `jquery.ui.datepicker-fr.js` 。所需的本地化文件需要包含在主要的日期选择器代码后面。每个本地化文件添加了它自己的设置到可用的本地化集合中，所有实例自动应用这些设置为默认设置。

`$.datepicker.regional` 属性保存了一个本地化数组，以语言代码作为前置，默认前置为 `""` ，表示英语。每个条目是一个带有下列属性的对象： `closeText` 、 `prevText` 、 `nextText` 、 `currentText` 、 `monthNames` 、 `monthNamesShort` 、 `dayNames` 、 `dayNamesShort` 、 `dayNamesMin` 、 `weekH` 、 `dateFormat` 、 `firstDay` 、 `isRTL` 、 `showMonthAfterYear` 和 `yearSuffix` 。

您可以通过下面代码恢复默认的本地化：

`$.datepicker.setDefaults( $.datepicker.regional[ "" ] );`

您可以通过下面代码覆盖一个特定地点的日期选择器：

`$( selector ).datepicker( $.datepicker.regional[ "fr" ] );`

主题化

日期选择器部件（Datepicker Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用日期选择器指定的样式，则可以使用下面的 CSS class 名称：

- `ui-datepicker` ：日期选择器的外层容器。如果日期选择器是内联的，该元素会另外带有一个 `ui-datepicker-inline` class。如果设置了 `isRTL` 选项，该元素会另外带有一个 `ui-datepicker-rtl` class。
  - `ui-datepicker-header` ：日期选择器的头部容器。
    - `ui-datepicker-prev` ：用于选择上一月的控件。
    - `ui-datepicker-next` ：用于选择下一月的控件。
    - `ui-datepicker-title` ：日期选择器包含月和年的标题容器。

        - ui-datepicker-month：月的文本显示，如果设置了 changeMonth 选项则显示 <select> 元素。
        - ui-datepicker-year：年的文本显示，如果设置了 changeYear 选项则显示 <select> 元素。
    - ui-datepicker-calendar：包含日历的表格。
        - ui-datepicker-week-end：周末的单元格。: Cells containing weekend days.
        - ui-datepicker-other-month：发生在某月但不是当前月天数的单元格。
        - ui-datepicker-unselectable：用户不可选择的单元格。
        - ui-datepicker-current-day：已选中日期的单元格。
        - ui-datepicker-today：当天日期的单元格。
    - ui-datepicker-buttonpane：当设置 showButtonPanel 选项时使用按钮面板（buttonpane）。
        - ui-datepicker-current：用于选择当天日期的按钮。

如果 numberOfMonths 选项用于显示多个月份，则使用一些额外的 class：

- ui-datepicker-multi：一个多月份日期选择器的最外层容器。该元素会根据要显示的月份个数另外带有 ui-datepicker-multi-2、ui-datepicker-multi-3 或 ui-datepicker-multi-4 class 名称。
    - ui-datepicker-group：分组内单独的选择器。该元素会根据它在分组中的位置另外带有 ui-datepicker-group-first、ui-datepicker-group-middle 或 ui-datepicker-group-last class 名称。

依赖

- UI 核心（UI Core）
- 特效核心（Effects Core）（可选的；当与 showAnim 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 change 事件。
- 不支持在 <input type="date"> 上创建日期选择器，因为会造成与本地选择器的 UI 冲突。

实例

一个简单的 jQuery UI 日期选择器（Datepicker）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>日期选择器部件（Datepicker Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="datepicker"></div>

<script>
$( "#datepicker" ).datepicker();
</script>

</body>
</html>
```

jQuery UI API - 对话框部件（Dialog Widget）

所属类别

小部件（Widgets）

用法

描述：在一个交互覆盖层中打开内容。

版本新增：1.0

对话框是一个悬浮窗口，包括一个标题栏和一个内容区域。对话框窗口可以移动，重新调整大小，默认情况下通过 'x' 图标关闭。

如果内容长度超过最大高度，一个滚动条会自动出现。

一个底部按钮栏和一个半透明的模式覆盖层是常见的添加选项。

焦点

当打开一个对话框时，焦点会自动移动到满足下面条件的第一个项目：

1. 带有 `autofocus` 属性的对话框内的第一个元素
2. 对话框内容内的第一个 `:tabbable` 元素
3. 对话框按钮面板内的第一个 `:tabbable` 元素
4. 对话框的关闭按钮
5. 对话框本身

当打开时，对话框部件（Dialog Widget）确保通过 tab 切换对话框内元素间的焦点，不包括对话框外的元素。模态对话框防止鼠标用户点击对话框外的元素。

当关闭对话框时，焦点自动返回到之前对话框打开时获得焦点的元素上。

隐藏关闭按钮

在一些情况下，您可能想要隐藏关闭按钮，例如，在按钮面板中已经有一个关闭按钮的情况。最好的解决方法是通过 CSS。作为实例，您可以定义一个简单的规则，比如：

```
.no-close .ui-dialog-titlebar-close {
  display: none;
}
```

然后，您可以添加 `no-close` class 到任意的对话框，用来隐藏关闭按钮：

```
$( "#dialog" ).dialog({
  dialogClass: "no-close",
  buttons: [
    {
      text: "OK",
      click: function() {
        $( this ).dialog( "close" );
      }
    }
  ]
});
```

主题化

对话框部件（Dialog Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用对话框指定的样式，则可以使用下面的 CSS class 名称：

- `ui-dialog`：对话框的外层容器。
  - `ui-dialog-titlebar`：包含对话框标题和关闭按钮的标题栏。
    - `ui-dialog-title`：对话框文本标题周围的容器。
    - `ui-dialog-titlebar-close`：对话框的关闭按钮。
  - `ui-dialog-content`：对话框内容周围的容器。这也是部件被实例化的元素。
  - `ui-dialog-buttonpane`：包含对话按钮的面板。只有当设置了 `buttons` 选项时才呈现。
    - `ui-dialog-buttonset`：按钮周围的容器。

此外，当设置了 `modal` 选项时，一个带有 `ui-widget-overlay` class 名称的元素被追加到 <body> 。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 按钮部件（Button Widget）
- 可拖拽小部件（Draggable Widget）（可选的；当与 `draggable` 选项一起使用时）
- 可调整尺寸小部件（Resizable Widget）（可选的；当与 `resizable` 选项一起使用时）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 对话框（Dialog）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>对话框部件（Dialog Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<button id="opener">打开对话框</button>
<div id="dialog" title="对话框标题">我是一个对话框</div>

<script>
$( "#dialog" ).dialog({ autoOpen: false });
$( "#opener" ).click(function() {
  $( "#dialog" ).dialog( "open" );
});
</script>

</body>
</html>
```

jQuery UI API - 菜单部件（Menu Widget）

所属类别

[小部件（Widgets）]

用法

描述：带有鼠标和键盘交互的用于导航的可主题化菜单。

版本新增：1.9

菜单可以用任何有效的标记创建，只要元素有严格的父/子关系且每个条目都有一个锚。最常用的元素是无序列表（ `<ul>` ）：

```
<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a>
    <ul>
      <li><a href="#">Item 3-1</a></li>
      <li><a href="#">Item 3-2</a></li>
      <li><a href="#">Item 3-3</a></li>
      <li><a href="#">Item 3-4</a></li>
      <li><a href="#">Item 3-5</a></li>
    </ul>
  </li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
</ul>
```

如果使用一个非 `<ul>` / `<li>` 的结构，为菜单和菜单条目使用相同的元素，请使用 `menus` 选项来区分两个元素，例如 `menus: "div.menuElement"` 。

可通过向元素添加 `ui-state-disabled` class 来禁用任何菜单条目。

图标

为了向菜单添加图标，请在标记中包含图标：

```
<ul id="menu">
  <li><a href="#"><span class="ui-icon ui-icon-disk"></span>Save</a
</ul>
```

菜单（Menu）会自动向无图标的条目添加必要的内边距。

分隔符

分隔符元素可通过包含未链接的菜单条目来创建，菜单条目只能是空格/破折号：

```
<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li>-</li>
  <li><a href="#">Item 2</a></li>
</ul>
```

键盘交互

- ENTER/SPACE：调用获得焦点的菜单项的动作，可能会打开一个子菜单。
- UP：移动教导到上一个菜单项。
- DOWN：移动教导到下一个菜单项。
- RIGHT：如果可用，则打开子菜单。
- LEFT：关闭当前子菜单，移动焦点到父菜单项。如果焦点不在子菜单上，则不进行任何操作。
- ESCAPE：关闭当前子菜单，移动焦点到父菜单项。如果焦点不在子菜单上，则不进行任何操作。

输入一个字母，移动焦点到以该字母开头的第一个条目。重复相同的字符会循环显示匹配的条目。在一个时间内输入更多的字符则匹配所输入的字符。

禁用项可获得键盘焦点，但是不允许任何交互。

主题化

菜单部件（Menu Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用菜单指定的样式，则可以使用下面的 CSS class 名称：

- `ui-menu`：菜单的外层容器。如果菜单包含图标，该元素会另外带有一个 `ui-menu-icons` class。
  - `ui-menu-item`：单个菜单项的容器。
    - `ui-menu-icon`：通过 `icons` 选项进行子菜单图标设置。
  - `ui-menu-divider`：菜单项之间的分隔符元素。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 菜单（Menu）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>菜单部件（Menu Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>
  .ui-menu {
    width: 200px;
  }
  </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<ul id="menu">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a>
    <ul>
      <li><a href="#">Item 3-1</a></li>
      <li><a href="#">Item 3-2</a></li>
      <li><a href="#">Item 3-3</a></li>
      <li><a href="#">Item 3-4</a></li>
      <li><a href="#">Item 3-5</a></li>
    </ul>
  </li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
</ul>

<script>
$( "#menu" ).menu();
</script>

</body>
</html>
```

jQuery UI API - 进度条部件（Progressbar Widget）

所属类别

小部件（Widgets）

用法

描述：显示一个确定的或不确定的进程状态。

版本新增：1.6

进度条被设计来显示进度的当前完成百分比。进度条通过 CSS 编码灵活调整大小，默认会缩放到适应父容器的大小。

一个确定的进度条只能在系统可以准确更新当前状态的情况下使用。一个确定的进度条不会从左向右填充，然后循环回到空 - 如果不能计算实际状态，则使用不确定的进度条以便提供用户反馈。

主题化

进度条部件（Progressbar Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用进度条指定的样式，则可以使用下面的 CSS class 名称：

- `ui-progressbar`：进度条的外层容器。该元素会为不确定的进度条另外添加一个 `ui-progressbar-indeterminate` class。
  - `ui-progressbar-value`：该元素代表进度条的填充部分。
    - `ui-progressbar-overlay`：用于为不确定的进度条显示动画的覆盖层。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 不确定的进度条（Indeterminate Progressbar）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>进度条部件（Progressbar Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="progressbar"></div>

<script>
$( "#progressbar" ).progressbar({
  value: false
});
</script>

</body>
</html>
```

jQuery UI API - 滑块部件（Slider Widget）

所属类别

小部件（Widgets）

用法

描述：拖动手柄来选择一个数值。

版本新增：1.5

jQuery UI 滑块（Slider）插件允许通过滑块进行选择。有各种不同的选项，比如多个手柄和范围。手柄可通过鼠标或箭头按键进行移动。

滑块部件（Slider Widget）会在初始化时创建带有 class `ui-slider-handle` 的手柄元素。您可以通过在初始化之前创建并追加元素，同时向元素添加 `ui-slider-handle` class 来指定自定义的手柄元素。它只会创建匹配 `value` / `values` 长度所需的数量的手柄。例如，如果您指定 `values: [ 1, 5, 18 ]`，且创建一个自定义手柄，插件将创建其他两个。

主题化

滑块部件（Slider Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用滑块指定的样式，则可以使用下面的 CSS class 名称：

- `ui-slider`：滑块控件的轨道。该元素会根据滑块的 `orientation` 另外带有一个 `ui-slider-horizontal` 或 `ui-slider-vertical` class。
  - `ui-slider-handle`：滑块手柄。
  - `ui-slider-range`：当设置 `range` 选项时使用的已选范围。如果 `range` 选项设置为 `"min"` 或 `"max"`，则该元素会分别另外带有一个 `ui-slider-range-min` 或 `ui-slider-range-max` class。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 鼠标交互（Mouse Interaction）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 滑块（Slider）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>滑块部件（Slider Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <style>#slider { margin: 10px; }    </style>
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="slider"></div>

<script>
$( "#slider" ).slider();
</script>

</body>
</html>
```

jQuery UI API - 旋转器部件（Spinner Widget）

所属类别

小部件（Widgets）

用法

描述：通过向上/向下按钮和箭头键处理，为输入数值增强文本输入功能。

版本新增：1.9

旋转器（Spinner），或数步进控件（number stepper widget），是用于处理各种数字输入的完美控件。它允许用户直接输入一个值，或通过键盘、鼠标、滚轮旋转改变一个已有的值。当与全球化（Globalize）结合时，您甚至可以旋转显示不同地区的货币和日期。

旋转器（Spinner）使用两个按钮将文本输入覆盖为当前值的递增值和递减值。旋转器增加了按键事件，以便可以用键盘完成相同的递增和递减。旋转器代表 全球化（Globalize）的数字格式和解析。

键盘交互

- UP：对值增加一步。
- DOWN：对值减少一步。
- PAGE UP：对值增加一页。
- PAGE DOWN：对值减少一页。

用鼠标点击旋转按钮后，焦点仍停留在文本域中。

当旋转器不是只读（ `<input readonly>` ）时，用户可以输入值，这可能会产生无效的值（小于最小值，大于最大值，增减错配，非数字输入）。当增减时，不管通过编程方式还是旋转按钮方式，值都会被强制为一个有效值（如需了解详情，请查看 `stepUp()` 和 `stepDown()` 的描述。

主题化

旋转器部件（Spinner Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用旋转器指定的样式，则可以使用下面的 CSS class 名称：

- `ui-spinner` ：旋转器的外层容器。
  - `ui-spinner-input` ：旋转器部件（Spinner Widget）实例化的 `<input>` 元素。
  - `ui-spinner-button` ：用于递增或递减旋转器值的按钮控件。向上按钮

会另外带有一个 `ui-spinner-up` class，向下按钮会另外带有一个 `ui-spinner-down` class。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 按钮部件（Button Widget）
- 全球化（Globalize）（外部的，可选的；当与 `culture` 和 `numberFormat` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。
- 该部件以编程方式操作元素的值，因此当元素的值改变时不会触发原生的 `change` 事件。
- 不支持在 `<input type="number">` 上创建选择器，因为会造成与本地旋转器器的 UI 冲突。

实例

普通的数字选择器。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>旋转器部件（Spinner Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<input id="spinner">

<script>
$( "#spinner" ).spinner();
</script>

</body>
</html>
```

jQuery UI API - 标签页部件（Tabs Widget）

所属类别

小部件（Widgets）

用法

描述：一种多面板的单内容区，每个面板与列表中的标题相关。

版本新增：1.0

标签页（Tabs）通常用于把内容分成多个部分，以便节省空间，就像折叠面板（accordion）一样。

标签页（Tabs）有一组必须使用的特定标记，以便标签页能正常工作：

- 标签页（Tabs）必须在一个有序的（ `<ol>` ）或无序的（ `<ul>` ）列表中
- 每个标签页的 "title" 必须在一个列表项（ `<li>` ）的内部，且必须用一个带有 `href` 属性的锚（ `<a>` ）包裹。
- 每个标签页面板可以是任意有效的元素，但是它必须带有一个 id，该 id 与相关标签页的锚中的哈希相对应。

每个标签页面板的内容可以在页面中定义好，或者可以通过 Ajax 加载。这两种方式都是基于与标签页相关的锚的 `href` 上自动处理的。默认情况下，标签页在点击时激活，但是通过 `event` 选项可以改变或覆盖事件。

下面是一些样品标记：

```
<div id="tabs">
  <ul>
    <li><a href="#fragment-1">一</a></li>
    <li><a href="#fragment-2">二</a></li>
    <li><a href="#fragment-3">三</a></li>
  </ul>
  <div id="fragment-1">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed d
  </div>
</div>
```

键盘交互

当焦点在标签页上时，下面的键盘命令可用：

- UP/LEFT：移动焦点到上一个标签页。如果在第一个标签页上，则移动焦点到最后一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- DOWN/RIGHT：移动焦点到下一个标签页。如果在最后一个标签页上，则移动焦点到第一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- HOME：移动焦点到第一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- END：移动焦点到最后一个标签页。在一个短暂的延迟后激活获得焦点的标签页。
- SPACE：激活与获得焦点的标签页相关的面板。
- ENTER：激活或切换与获得焦点的标签页相关的面板。
- ALT+PAGE UP：移动焦点到上一个标签页，并立即激活。
- ALT+PAGE DOWN：移动焦点到下一个标签页，并立即激活。

当焦点在面板上时，下面的键盘命令可用：

- CTRL+UP：移动焦点到相关的标签页。
- ALT+PAGE UP：移动焦点到上一个标签页，并立即激活。
- ALT+PAGE DOWN：移动焦点到下一个标签页，并立即激活。

主题化

标签页部件（Tabs Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用标签页指定的样式，则可以使用下面的 CSS class 名称：

- `ui-tabs` ：标签页的外层容器。当设置了 `collapsible` 选项时，该元素会另外带有一个 `ui-tabs-collapsible` class。

  - `ui-tabs-nav`：标签页列表。
    - 导航中激活的列表项会带有一个 `ui-tabs-active` class。内容通过 Ajax 调用加载的列表项会带有一个 `ui-tabs-loading` class。
      - `ui-tabs-anchor`：用于切换面板的锚。
  - `ui-tabs-panel`：与标签页相关的面板。只有与其对应的标签页激活时才可见。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

一个简单的 jQuery UI 标签页（Tabs）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>标签页部件（Tabs Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#fragment-1"><span>一</span></a></li>
    <li><a href="#fragment-2"><span>二</span></a></li>
    <li><a href="#fragment-3"><span>三</span></a></li>
  </ul>
  <div id="fragment-1">
    <p>第一个标签是默认激活的：</p>
    <code>$( "#tabs" ).tabs(); </code>
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed c
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed c
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed c
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed c
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed c
  </div>
</div>

<script>
$( "#tabs" ).tabs();
</script>

</body>
</html>
```

jQuery UI API - 工具提示框部件（Tooltip Widget）

所属类别

小部件（Widgets）

用法

描述：可自定义的、可主题化的工具提示框，替代原生的工具提示框。

版本新增：1.9

工具提示框（Tooltip）取代了原生的工具提示框，让它们可主题化，也允许进行各种自定义：

- 显示不仅仅是标题的其他内容，就如内联的脚注或通过 Ajax 检索的额外内容。
- 自定义定位，例如，在元素上居中工具提示框。
- 添加额外的样式来定制警告或错误区域的外观。

默认使用一个渐变的动画来显示和隐藏工具提示框，这种外观与简单的切换可见度相比更具灵性。这可以通过 `show` 和 `hide` 选项进行定制。

`items` 和 `content` 选项需要保持同步。如果您改变了其中一个，您需要同时改变另一个。

在一般情况下，禁用的元素不会触发任何 DOM 事件。因此，适当地控制禁用元素的工具提示框是不可能的，因为我们需要监听事件来决定何时显示和隐藏工具提示框。这就导致 jQuery UI 不能保证对附加到禁用元素上的工具提示框任何层次上的支持。这意味着如果您需要在禁用元素上进行提示，您可能需要使用一个原生的提示框和 jQuery UI 工具提示框的混合物。

主题化

工具提示框部件（Tooltip Widget）使用 jQuery UI CSS 框架 来定义它的外观和感观的样式。如果需要使用工具提示框指定的样式，则可以使用下面的 CSS class 名称：

- `ui-tooltip` ：工具提示框的外层容器。
    - `ui-tooltip-content` ：工具提示框的内容。

依赖

- UI 核心（UI Core）
- 部件库（Widget Factory）
- 定位（Position）
- 特效核心（Effects Core）（可选的；当与 `show` 和 `hide` 选项一起使用时）

附加说明

- 该部件要求一些功能性的 CSS，否则将无法工作。如果您创建了一个自定义的主题，请使用小部件指定的 CSS 文件作为起点。

实例

使用带有 title 属性的所有元素的事件代理，在文档上创建一个工具提示框（Tooltip）。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>工具提示框部件（Tooltip Widget）演示</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/s
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
</head>
<body>

<p>
  <a href="#" title="锚描述">锚文本</a>
  <input title="输入帮助">
</p>
<script>
  $( document ).tooltip();
</script>

</body>
</html>
```

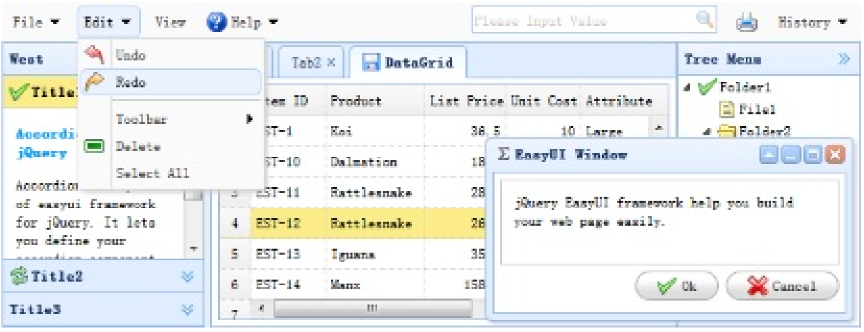
W3School jQuery Easy UI教程

来源：jQuery Easy UI教程

整理：飞龙

jQuery EasyUI 简介

jQuery EasyUI 是一个基于 jQuery 的框架，集成了各种用户界面插件。



什么是 jQuery EasyUI

jQuery EasyUI 框架提供了创建网页所需的一切，帮助您轻松建立站点。

- easyui 是一个基于 jQuery 的框架，集成了各种用户界面插件。
- easyui 提供建立现代化的具有交互性的 javascript 应用的必要的功能。
- 使用 easyui，您不需要写太多 javascript 代码，一般情况下您只需要使用一些 html 标记来定义用户界面。
- HTML 网页的完整框架。
- easyui 节省了开发产品的时间和规模。
- easyui 非常简单，但是功能非常强大。



jQuery EasyUI 下载

您可以从 http://www.jeasyui.com/download/index.php 上下载您需要的 jQuery EasyUI 版本。

轻松使用 jQuery 和 HTML5

jQuery EasyUI 提供易于使用的组件，它使 Web 开发人员快速地在流行的 jQuery 核心和 HTML5 上建立程序页面。 这些功能使您的应用适合今天的网络。 有两个方法声明的 UI 组件:

1. 直接在 HTML 声明组件。

```
<div class="easyui-dialog" style="width:400px;height:200px"
    data-options="title:'My Dialog',collapsible:true,iconCls:'icon-
        dialog content.
</div>
```

2. 编写 JavaScript 代码来创建组件。

```
<input id="cc" style="width:200px" />
```

```
$('#cc').combobox({
    url: ...,
    required: true,
    valueField: 'id',
    textField: 'text'
});
```

jQuery EasyUI 应用

jQuery EasyUI 应用 - 创建 CRUD 应用

数据收集并妥善管理数据是网络应用共同的必要。CRUD 允许我们生成页面列表，并编辑数据库记录。本教程将向你演示如何使用 jQuery EasyUI 框架实现一个 CRUD DataGrid。

我们将使用下面的插件：

- datagrid：向用户展示列表数据。
- dialog：创建或编辑一条单一的用户信息。
- form：用于提交表单数据。
- messager：显示一些操作信息。

步骤 1：准备数据库

我们将使用 MySql 数据库来存储用户信息。创建数据库和 'users' 表。

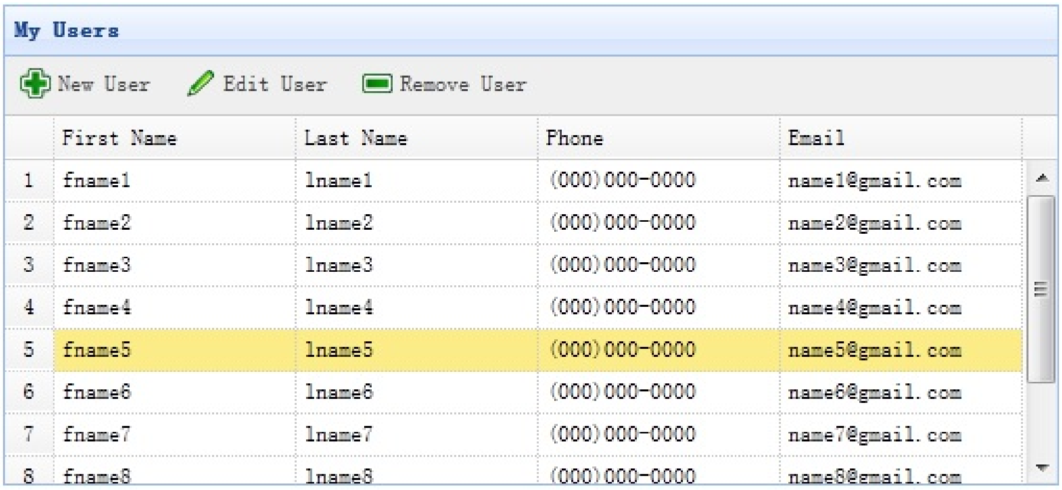
id	firstname	lastname	phone	email
3	fname1	lname1	(000)000-0000	name1@gmail.com
4	fname2	lname2	(000)000-0000	name2@gmail.com
5	fname3	lname3	(000)000-0000	name3@gmail.com
7	fname4	lname4	(000)000-0000	name4@gmail.com
8	fname5	lname5	(000)000-0000	name5@gmail.com
9	fname6	lname6	(000)000-0000	name6@gmail.com
10	fname7	lname7	(000)000-0000	name7@gmail.com
11	fname8	lname8	(000)000-0000	name8@gmail.com
12	fname9	lname9	(000)000-0000	name9@gmail.com
13	fname10	lname10	(000)000-0000	name10@gmail.com

步骤 2：创建 DataGrid 来显示用户信息

创建没有 javascript 代码的 DataGrid。

```
<table id="dg" title="My Users" class="easyui-datagrid" style="widt
        url="get_users.php"
        toolbar="#toolbar"
        rownumbers="true" fitColumns="true" singleSelect="true">
    <thead>
        <tr>
            <th field="firstname" width="50">First Name</th>
            <th field="lastname" width="50">Last Name</th>
            <th field="phone" width="50">Phone</th>
            <th field="email" width="50">Email</th>
        </tr>
    </thead>
</table>
<div id="toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain=
    <a href="#" class="easyui-linkbutton" iconCls="icon-edit" plair
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" pla
</div>
```

我们不需要写任何的 javascript 代码，就能向用户显示列表，如下图所示：



DataGrid 使用 'url' 属性，并赋值为 'get_users.php'，用来从服务器检索数据。

get_users.php 文件的代码

```
$rs = mysql_query('select * from users');
$result = array();
while($row = mysql_fetch_object($rs)){
    array_push($result, $row);
}

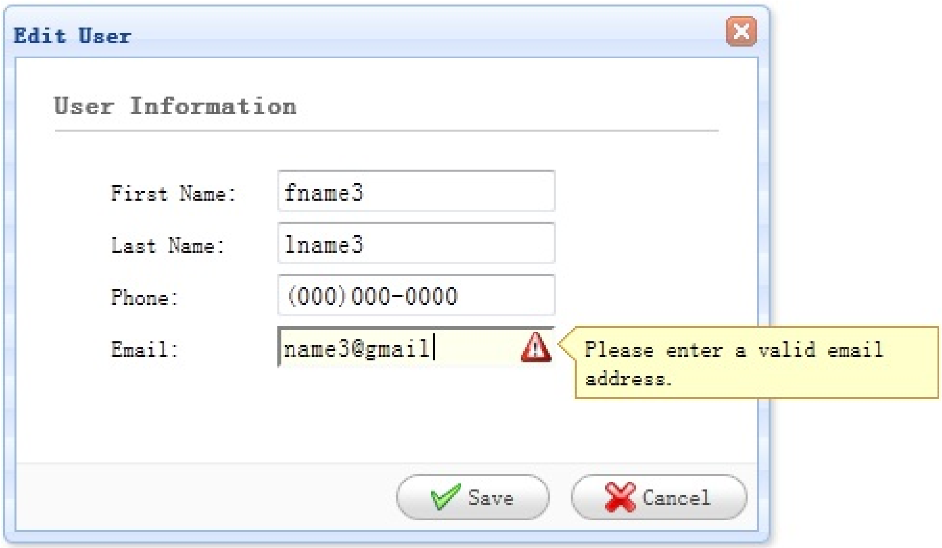
echo json_encode($result);
```

步骤 3：创建表单对话框

我们使用相同的对话框来创建或编辑用户。

```
<div id="dlg" class="easyui-dialog" style="width:400px;height:280px
        closed="true" buttons="#dlg-buttons">
    <div class="ftitle">User Information</div>
    <form id="fm" method="post">
        <div class="fitem">
            <label>First Name:</label>
            <input name="firstname" class="easyui-validatebox" requ
        </div>
        <div class="fitem">
            <label>Last Name:</label>
            <input name="lastname" class="easyui-validatebox" requi
        </div>
        <div class="fitem">
            <label>Phone:</label>
            <input name="phone">
        </div>
        <div class="fitem">
            <label>Email:</label>
            <input name="email" class="easyui-validatebox" validTyp
        </div>
    </form>
</div>
<div id="dlg-buttons">
    <a href="#" class="easyui-linkbutton" iconCls="icon-ok" onclick
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel" onc
</div>
```

这个对话框已经创建，也没有任何的 javascript 代码:。



步骤 4：实现创建和编辑用户

当创建用户时，打开一个对话框并清空表单数据。

```
function newUser(){
    $('#dlg').dialog('open').dialog('setTitle','New User');
    $('#fm').form('clear');
    url = 'save_user.php';
}
```

当编辑用户时，打开一个对话框并从 datagrid 选择的行中加载表单数据。

```
var row = $('#dg').datagrid('getSelected');
if (row){
    $('#dlg').dialog('open').dialog('setTitle','Edit User');
    $('#fm').form('load',row);
    url = 'update_user.php?id='+row.id;
}
```

'url' 存储着当保存用户数据时表单回传的 URL 地址。

步骤 5：保存用户数据

我们使用下面的代码保存用户数据：

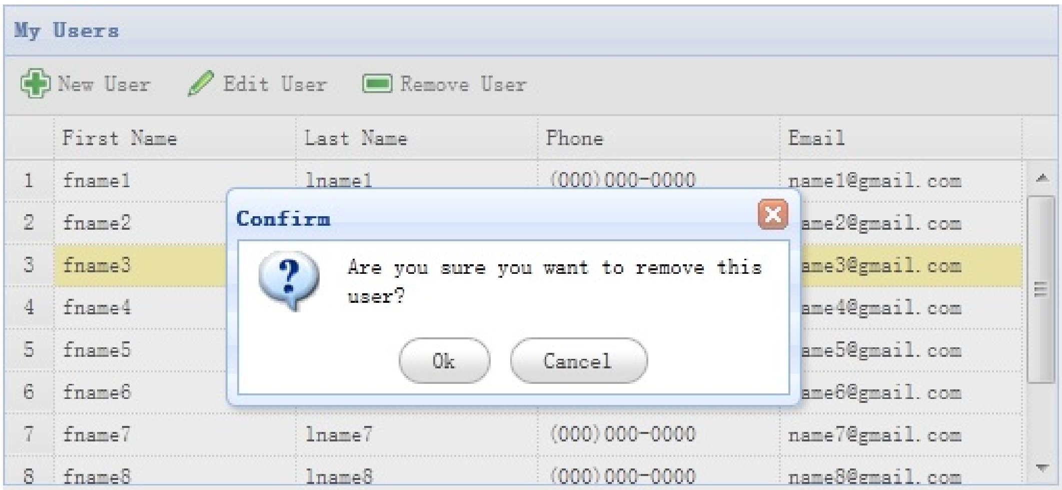
```
function saveUser(){
    $('#fm').form('submit',{
        url: url,
        onSubmit: function(){
            return $(this).form('validate');
        },
        success: function(result){
            var result = eval('('+result+')');
            if (result.errorMsg){
                $.messager.show({
                    title: 'Error',
                    msg: result.errorMsg
                });
            } else {
                $('#dlg').dialog('close');        // close the dial
                $('#dg').datagrid('reload');    // reload the user
            }
        }
    });
}
```

提交表单之前，'onSubmit' 函数将被调用，该函数用来验证表单字段值。当表单字段值提交成功，关闭对话框并重新加载 datagrid 数据。

步骤 6：删除一个用户

我们使用下面的代码来移除一个用户：

```
function destroyUser(){
    var row = $('#dg').datagrid('getSelected');
    if (row){
        $.messager.confirm('Confirm','Are you sure you want to dest
            if (r){
                $.post('destroy_user.php',{id:row.id},function(resu
                    if (result.success){
                        $('#dg').datagrid('reload');    // reload t
                    } else {
                        $.messager.show({    // show error message
                            title: 'Error',
                            msg: result.errorMsg
                        });
                    }
                },'json');
            }
        });
    }
}
```



移除一行之前,我们将显示一个确认对话框让用户决定是否真的移除该行数据。当移除数据成功之后，调用 'reload' 方法来刷新 datagrid 数据。

步骤 7：运行代码

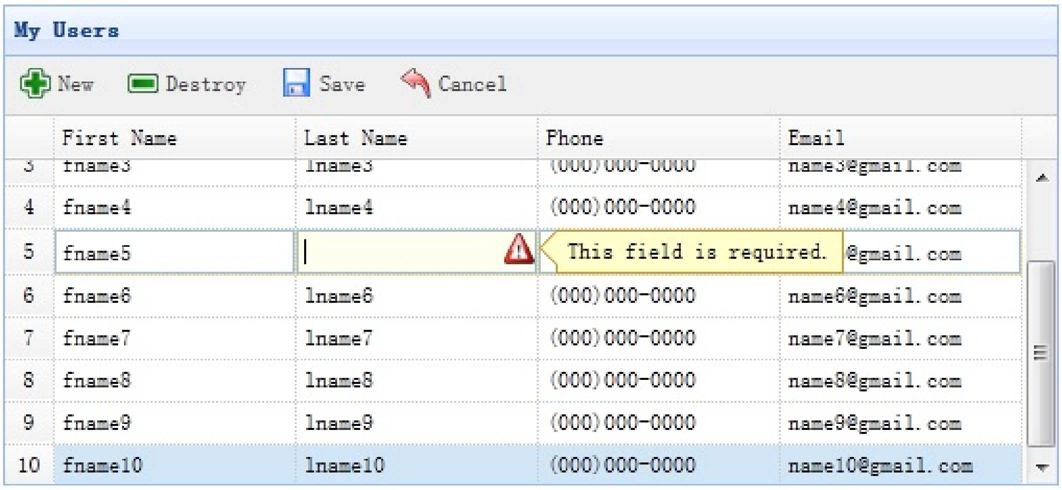
开启 MySQL，在浏览器运行代码。

下载 jQuery EasyUI 实例

jeasyui-app-crud1.zip

jQuery EasyUI 应用 - 创建 CRUD 数据网格（DataGrid）

在上一章节中，我们使用对话框（dialog）组件创建了 CRUD 应用来创建和编辑用户信息。本教程我们将告诉您如何创建一个 CRUD 数据网格（DataGrid）。 我们将使用 可编辑的数据网格（DataGrid）插件 来完成这些 CRUD 操作动作。



步骤 1：在 HTML 标签中定义数据网格（DataGrid）

```
<table id="dg" title="My Users" style="width:550px;height:250px"
        toolbar="#toolbar" idField="id"
        rownumbers="true" fitColumns="true" singleSelect="true">
    <thead>
        <tr>
            <th field="firstname" width="50" editor="{type:'validat
            <th field="lastname" width="50" editor="{type:'validate
            <th field="phone" width="50" editor="text">Phone</th>
            <th field="email" width="50" editor="{type:'validatebox
        </tr>
    </thead>
</table>
<div id="toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain=
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" pla
    <a href="#" class="easyui-linkbutton" iconCls="icon-save" plain
    <a href="#" class="easyui-linkbutton" iconCls="icon-undo" plain
</div>
```

步骤 2：使用可编辑的数据网格（DataGrid）

```
$('#dg').edatagrid({
    url: 'get_users.php',
    saveUrl: 'save_user.php',
    updateUrl: 'update_user.php',
    destroyUrl: 'destroy_user.php'
});
```

我们应该提供 'url'、'saveUrl'、'updateUrl' 和 'destroyUrl' 属性来编辑数据网格（DataGrid）：

- url：从服务器端检索用户数据。
- saveUrl：保存一个新的用户数据。
- updateUrl：更新一个已存在的用户数据。
- destroyUrl：删除一个已存在的用户数据。

步骤 3：写服务器处理代码

保存一个新的用户（save_user.php）：

```
$firstname = $_REQUEST['firstname'];
$lastname = $_REQUEST['lastname'];
$phone = $_REQUEST['phone'];
$email = $_REQUEST['email'];

include 'conn.php';

$sql = "insert into users(firstname,lastname,phone,email) values('$
@mysql_query($sql);
echo json_encode(array(
    'id' => mysql_insert_id(),
    'firstname' => $firstname,
    'lastname' => $lastname,
    'phone' => $phone,
    'email' => $email
));
```

更新一个已存在用户（update_user.php）：

```
$id = intval($_REQUEST['id']);
$firstname = $_REQUEST['firstname'];
$lastname = $_REQUEST['lastname'];
$phone = $_REQUEST['phone'];
$email = $_REQUEST['email'];

include 'conn.php';

$sql="update users set firstname='$firstname',lastname='$lastname',
@mysql_query($sql);
echo json_encode(array(
    'id' => $id,
    'firstname' => $firstname,
    'lastname' => $lastname,
    'phone' => $phone,
    'email' => $email
));
```

删除一个已存在用户（destroy_user.php）：

```
$id = intval($_REQUEST['id']);

include 'conn.php';

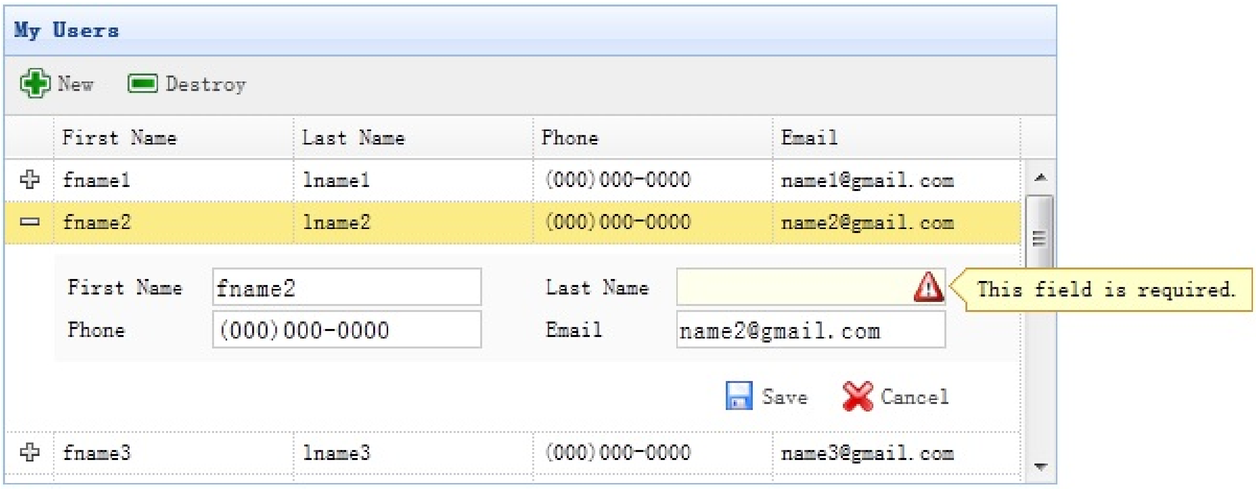
$sql = "delete from users where id=$id";
@mysql_query($sql);
echo json_encode(array('success'=>true));
```

下载 jQuery EasyUI 实例

jeasyui-app-crud2.zip

jQuery EasyUI 应用 - 创建展开行明细编辑表单的 CRUD 应用

当切换数据网格视图（datagrid view）到 'detailview'，用户可以展开一行来显示一些行的明细在行下面。这个功能允许您为防止在明细行面板（panel）中的编辑表单（form）提供一些合适的布局（layout）。在本教程中，我们使用数据网格（datagrid）组件来减小编辑表单（form）所占据空间。



步骤 1：在 HTML 标签中定义数据网格（DataGrid）

```
<table id="dg" title="My Users" style="width:550px;height:250px"
        url="get_users.php"
        toolbar="#toolbar"
        fitColumns="true" singleSelect="true">
    <thead>
        <tr>
            <th field="firstname" width="50">First Name</th>
            <th field="lastname" width="50">Last Name</th>
            <th field="phone" width="50">Phone</th>
            <th field="email" width="50">Email</th>
        </tr>
    </thead>
</table>
<div id="toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain=
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" pla
</div>
```

步骤 2：为数据网格（DataGrid）应用明细视图

```
$('#dg').datagrid({
    view: detailview,
    detailFormatter:function(index,row){
        return '<div class="ddv"></div>';
    },
    onExpandRow: function(index,row){
        var ddv = $(this).datagrid('getRowDetail',index).find('div
        ddv.panel({
            border:false,
            cache:true,
            href:'show_form.php?index='+index,
            onLoad:function(){
                $('#dg').datagrid('fixDetailRowHeight',index);
                $('#dg').datagrid('selectRow',index);
                $('#dg').datagrid('getRowDetail',index).find('form
            }
        });
        $('#dg').datagrid('fixDetailRowHeight',index);
    }
});
```

为了为数据网格（DataGrid）应用明细视图，在 html 页面头部引入 'datagrid-detailview.js' 文件。

我们使用 'detailFormatter' 函数来生成行明细内容。 在这种情况下，我们返回一个用于放置编辑表单（form）的空的 <div>。 当用户点击行展开按钮（'+'）时，'onExpandRow' 事件将被触发，我们将通过 ajax 加载编辑表单（form）。 调用 'getRowDetail' 方法来得到行明细容器，所以我们能查找到行明细面板（panel）。 在行明细中创建面板（panel），加载从 'show_form.php' 返回的编辑表单（form）。

步骤 3：创建编辑表单（Form）

编辑表单（form）是从服务器加载的。

show_form.php

```
<form method="post">
    <table class="dv-table" style="width:100%;background:#fafafa;pa
        <tr>
            <td>First Name</td>
            <td><input name="firstname" class="easyui-validatebox" 
            <td>Last Name</td>
            <td><input name="lastname" class="easyui-validatebox" r
        </tr>
        <tr>
            <td>Phone</td>
            <td><input name="phone"></input></td>
            <td>Email</td>
            <td><input name="email" class="easyui-validatebox" vali
        </tr>
    </table>
    <div style="padding:5px 0;text-align:right;padding-right:30px">
        <a href="#" class="easyui-linkbutton" iconCls="icon-save" p
        <a href="#" class="easyui-linkbutton" iconCls="icon-cancel"
    </div>
</form>
```

步骤 4：保存或取消编辑

调用 'saveItem' 函数来保存一个用户或者调用 'cancelItem' 函数来取消编辑。

```
function saveItem(index){
    var row = $('#dg').datagrid('getRows')[index];
    var url = row.isNewRecord ? 'save_user.php' : 'update_user.php?
    $('#dg').datagrid('getRowDetail',index).find('form').form('subm
        url: url,
        onSubmit: function(){
            return $(this).form('validate');
        },
        success: function(data){
            data = eval('('+data+')');
            data.isNewRecord = false;
            $('#dg').datagrid('collapseRow',index);
            $('#dg').datagrid('updateRow',{
                index: index,
                row: data
            });
        }
    });
}
```

决定要回传哪一个 URL，然后查找表单（form）对象，并调用 'submit' 方法来提交表单（form）数据。当保存数据成功时，折叠并更新行数据。

```
function cancelItem(index){
    var row = $('#dg').datagrid('getRows')[index];
    if (row.isNewRecord){
        $('#dg').datagrid('deleteRow',index);
    } else {
        $('#dg').datagrid('collapseRow',index);
    }
}
```

当取消编辑动作时，如果该行是新行而且还没有保存，直接删除该行，否则折叠该行。

下载 jQuery EasyUI 实例

jeasyui-app-crud3.zip

jQuery EasyUI 应用 - 创建 RSS Feed 阅读器

在本教程中，我们将通过 jQuery EasyUI 框架创建一个 RSS 阅读器。



我们将使用以下插件:

- layout：创建应用的用户界面。
- datagrid：显示 RSS Feed 列表。
- tree：显示 feed 频道。

步骤 1：创建布局（Layout）

```
&lt;body class="easyui-layout"&gt;
    <div region="north" border="false" class="rtitle">
        jQuery EasyUI RSS Reader Demo
    </div>
    <div region="west" title="Channels Tree" split="true" border="f
        <ul id="t-channels" url="data/channels.json"></ul>
    </div>
    <div region="center" border="false">
        <div class="easyui-layout" fit="true">
            <div region="north" split="true" border="false" style='
                <table id="dg"
                        url="get_feed.php" border="false" rownumber
                        fit="true" fitColumns="true" singleSelect='
                    <thead>
                        <tr>
                            <th field="title" width="100">Title</th
                            <th field="description" width="200">Des
                            <th field="pubdate" width="80">Publish
                        </tr>
                    </thead>
                </table>
            </div>
            <div region="center" border="false" style="overflow:hic
                <iframe id="cc" scrolling="auto" frameborder="0" st
            </div>
        </div>
    </div>
&lt;/body&gt;
```

步骤 2：数据网格（DataGrid）处理事件

在这里我们要处理一些由用户触发的事件。

```
$('#dg').datagrid({
    onSelect: function(index,row){
        $('#cc').attr('src', row.link);
    },
    onLoadSuccess:function(){
        var rows = $(this).datagrid('getRows');
        if (rows.length){
            $(this).datagrid('selectRow',0);
        }
    }
});
```

本实例使用 'onSelect' 事件来显示 feed 的内容，使用 'onLoadSuccess' 事件来选择第一行。

步骤 3：树形菜单（Tree）处理事件

当树形菜单（Tree）数据已经加载，我们需要选择第一个叶子节点，调用 'select' 方法来选择该节点。 使用 'onSelect' 事件来得到已选择的节点，这样我们就能得到对应的 'url' 值。 最后我们调用数据网格（DataGrid） 的 'load' 方法来刷新 feed 列表数据。

```
$('#t-channels').tree({
    onSelect: function(node){
        var url = node.attributes.url;
        $('#dg').datagrid('load',{
            url: url
        });
    },
    onLoadSuccess:function(node,data){
        if (data.length){
            var id = data[0].children[0].children[0].id;
            var n = $(this).tree('find', id);
            $(this).tree('select', n.target);
        }
    }
});
```

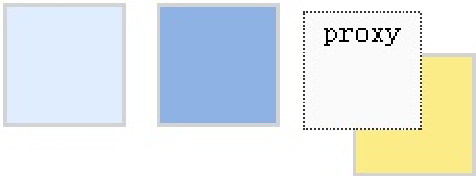
下载 jQuery EasyUI 实例

jeasyui-app-rssreader.zip

jQuery EasyUI 拖放

jQuery EasyUI 拖放 - 基本的拖动和放置

本教程向您展示如何使 HTML 元素可拖动，在本例中，我们将创建三个 DIV 元素然后启用他们的拖动和放置。



首先，我们创建三个 <div> 元素：

```
<div id="dd1" class="dd-demo"></div>
<div id="dd2" class="dd-demo"></div>
<div id="dd3" class="dd-demo"></div>
```

对于第一个 <div> 元素，我们通过默认值让其可以拖动。

```
$('#dd1').draggable();
```

对于第二个 <div> 元素，我们通过创建一个克隆（clone）了原来元素的代理（proxy）让其可以拖动。

```
$('#dd2').draggable({
    proxy:'clone'
});
```

对于第三个 <div> 元素，我们通过创建自定义代理（proxy）让其可以拖动。

```
$('#dd3').draggable({
    proxy:function(source){
        var p = $('<div class="proxy">proxy</div>');
        p.appendTo('body');
        return p;
    }
});
```

下载 jQuery EasyUI 实例

jeasyui-dd-basic.zip

jQuery EasyUI 拖放 - 创建拖放的购物车

如果您能够通过您的 Web 应用简单地实现拖动和放置，您就会知道一些特别的东西。通过 jQuery EasyUI，我们在 Web 应用中可以简单地实现拖放功能。

在本教程中，我们将向您展示如何创建一个启用用户拖动和放置用户想买的商品的购物车页面。购物篮中的物品和价格将更新。



显示页面上的商品

```
<ul class="products">
    <li>
        <a href="#" class="item">
            <img src="images/shirt1.gif"/>
            <div>
                <p>Balloon</p>
                <p>Price:$25</p>
            </div>
        </a>
    </li>
    <li>
        <a href="#" class="item">
            <img src="images/shirt2.gif"/>
            <div>
                <p>Feeling</p>
                <p>Price:$25</p>
            </div>
        </a>
    </li>
    <!-- other products -->
</ul>
```

正如您所看到的上面的代码，我们添加一个包含一些 <li> 元素的 <ul> 元素来显示商品。所有商品都有名字和价格属性，它们包含在 <p> 元素中。

创建购物车

```
<div class="cart">
    <h1>Shopping Cart</h1>
    <table id="cartcontent" style="width:300px;height:auto;">
        <thead>
            <tr>
                <th field="name" width=140>Name</th>
                <th field="quantity" width=60 align="right">Qua
                <th field="price" width=60 align="right">Price<
            </tr>
        </thead>
    </table>
    <p class="total">Total: $0</p>
    <h2>Drop here to add to cart</h2>
</div>
```

我们使用数据网格（datagrid）来显示购物篮中的物品。

拖动克隆的商品

```
$('.item').draggable({
    revert:true,
    proxy:'clone',
    onStartDrag:function(){
        $(this).draggable('options').cursor = 'not-allowed';
        $(this).draggable('proxy').css('z-index',10);
    },
    onStopDrag:function(){
        $(this).draggable('options').cursor='move';
    }
});
```

请注意，我们把 draggable 属性的值从 'proxy' 设置为 'clone'，所以拖动元素将由克隆产生。

放置选择商品到购物车中

```
$('.cart').droppable({
    onDragEnter:function(e,source){
        $(source).draggable('options').cursor='auto';
    },
    onDragLeave:function(e,source){
        $(source).draggable('options').cursor='not-allowed';
    },
    onDrop:function(e,source){
        var name = $(source).find('p:eq(0)').html();
        var price = $(source).find('p:eq(1)').html();
        addProduct(name, parseFloat(price.split('$')[1]));
    }
});
var data = {"total":0,"rows":[]};
var totalCost = 0;
function addProduct(name,price){
    function add(){
        for(var i=0; i<data.total; i++){
            var row = data.rows[i];
            if (row.name == name){
                row.quantity += 1;
                return;
            }
        }
        data.total += 1;
        data.rows.push({
            name:name,
            quantity:1,
            price:price
        });
    }
    add();
    totalCost += price;
    $('#cartcontent').datagrid('loadData', data);
    $('div.cart .total').html('Total: $'+totalCost);
}
```

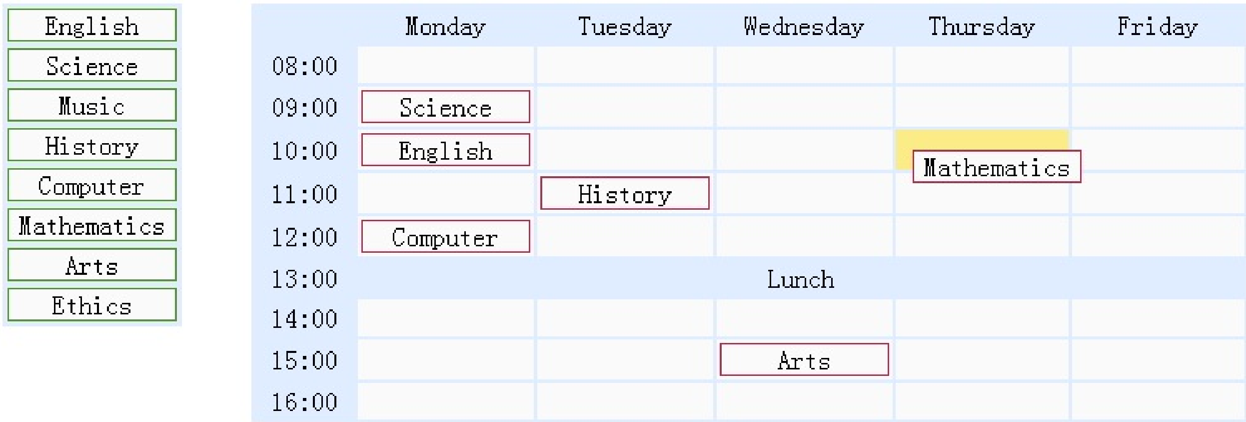
每当放置商品的时候，我们首先得到商品名称和价格，然后调用 'addProduct' 函数来更新购物篮。

下载 jQuery EasyUI 实例

jeasyui-dd-shopping.zip

jQuery EasyUI 拖放 - 创建学校课程表

本教程将向您展示如何使用 jQuery EasyUI 创建一个学校课程表。 我们将创建两个表格：在左侧显示学校科目，在右侧显示时间表。 您可以拖动学校科目并放置到时间表单元格上。 学校科目是一个 <div class="item"> 元素，时间表单元格是一个 <td class="drop"> 元素。



显示学校科目

```
<div class="left">
    <table>
        <tr>
            <td><div class="item">English</div></td>
        </tr>
        <tr>
            <td><div class="item">Science</div></td>
        </tr>
        <!-- other subjects -->
    </table>
</div>
```

显示时间表

```
<div class="right">
    <table>
        <tr>
            <td class="blank"></td>
            <td class="title">Monday</td>
            <td class="title">Tuesday</td>
            <td class="title">Wednesday</td>
            <td class="title">Thursday</td>
            <td class="title">Friday</td>
        </tr>
        <tr>
            <td class="time">08:00</td>
            <td class="drop"></td>
            <td class="drop"></td>
            <td class="drop"></td>
            <td class="drop"></td>
            <td class="drop"></td>
        </tr>
        <!-- other cells -->
    </table>
</div>
```

拖动在左侧的学校科目

```
$('.left .item').draggable({
    revert:true,
    proxy:'clone'
});
```

放置学校科目在时间表单元格上

```
    $('.right td.drop').droppable({
        onDragEnter:function(){
            $(this).addClass('over');
        },
        onDragLeave:function(){
            $(this).removeClass('over');
        },
        onDrop:function(e,source){
            $(this).removeClass('over');
            if ($(source).hasClass('assigned')){
                $(this).append(source);
            } else {
                var c = $(source).clone().addClass('assigned');
                $(this).empty().append(c);
                c.draggable({
                    revert:true
                });
            }
        }
    });
```

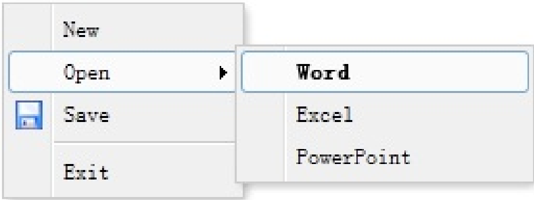
正如您所看到的上面的代码，当用户拖动在左侧的学校科目并放置到时间表单元格中时，onDrop 回调函数将被调用。我们克隆从左侧拖动的源元素并把它附加到时间表单元格上。 当把学校科目从时间表的某个单元格拖动到其他单元格，只需简单地移动它即可。

下载 jQuery EasyUI 实例

jeasyui-dd-timetable.zip

jQuery EasyUI 菜单与按钮

jQuery EasyUI 菜单与按钮 - 创建简单的菜单



菜单（Menu）定义在一些 DIV 标记中，如下所示：

```
<div id="mm" class="easyui-menu" style="width:120px;">
    <div onclick="javascript:alert('new')">New</div>
    <div>
        <span>Open</span>
        <div style="width:150px;">
            <div><b>Word</b></div>
            <div>Excel</div>
            <div>PowerPoint</div>
        </div>
    </div>
    <div icon="icon-save">Save</div>
    <div class="menu-sep"></div>
    <div>Exit</div>
</div>
```

当菜单创建之后是不显示的，调用 'show' 方法显示它或者调用 'hide' 方法隐藏它：

```
$('#mm').menu('show', {
  left: 200,
  top: 100
});
```

下载 jQuery EasyUI 实例

jeasyui-mb-menu.zip

jQuery EasyUI 菜单与按钮 - 创建链接按钮（Link Button）

通常情况下，使用 <button> 元素来创建按钮，而链接按钮（Link Button）则是使用 <a> 元素来创建的。所以实际上一个链接按钮（Link Button）就是一个显示为按钮样式的 <a> 元素。





为了创建链接按钮（Link Button），所有您需要做的就是添加一个名为 'easyui-linkbutton' 的 class 属性到 <a> 元素：

```
<div style="padding:5px;background:#fafafa;width:500px;border:1
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel"
    <a href="#" class="easyui-linkbutton" iconCls="icon-reload"
    <a href="#" class="easyui-linkbutton" iconCls="icon-search"
    <a href="#" class="easyui-linkbutton">text button</a>
    <a href="#" class="easyui-linkbutton" iconCls="icon-print">
</div>

<div style="padding:5px;background:#fafafa;width:500px;border:1
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true">text but
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    <a href="#" class="easyui-linkbutton" plain="true" iconCls=
</div>
```

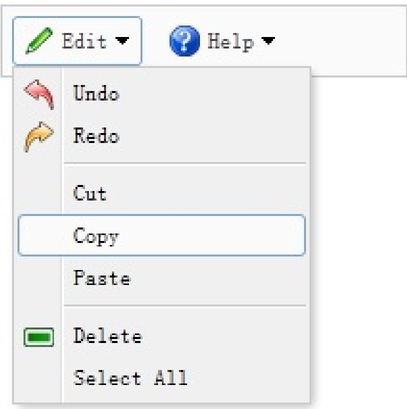
正如您所看到的，iconCls 属性是一个 icon 的 CSS class 样式，它在按钮上显示一个 icon 图片。

有时候您需要禁用链接按钮（Link Button）或者启用它，下面的代码演示了如何禁用一个链接按钮（Link Button）：

```
$(selector).linkbutton('disable');    // call the 'disable' met
```

jQuery EasyUI 菜单与按钮 - 创建菜单按钮（Menu Button）

菜单按钮（Menu Button）包含一个按钮（button）和一个菜单（menu）组件，当点击或移动鼠标到按钮上，将显示一个对应的菜单。



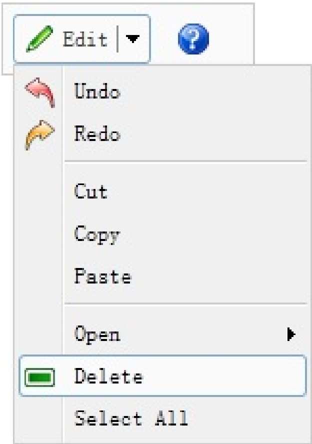
为了定义一个菜单按钮（Menu Button），您应该定义一个链接按钮（Link Button）和一个菜单（menu），下面是一个实例：

```
    <div style="background:#fafafa;padding:5px;width:200px;border:1
        <a href="#" class="easyui-menubutton" menu="#mm1" iconCls='
        <a href="#" class="easyui-menubutton" menu="#mm2" iconCls='
    </div>
    <div id="mm1" style="width:150px;">
        <div iconCls="icon-undo">Undo</div>
        <div iconCls="icon-redo">Redo</div>
        <div class="menu-sep"></div>
        <div>Cut</div>
        <div>Copy</div>
        <div>Paste</div>
        <div class="menu-sep"></div>
        <div iconCls="icon-remove">Delete</div>
        <div>Select All</div>
    </div>
    <div id="mm2" style="width:100px;">
        <div>Help</div>
        <div>Update</div>
        <div>About</div>
    </div>
```

现在已经定义好了一个菜单按钮（Menu Button），您不需要写任何的 javascript 代码。

jQuery EasyUI 菜单与按钮 - 创建分割按钮（Split Button）

分割按钮（Split Button）包含一个链接按钮（Link Button）和一个菜单（Menu）。当用户点击或者鼠标悬停在向下箭头区域，将会显示一个对应的菜单。本实例演示了如何创建和使用分割按钮（Split Button）。



我们创建一个分割按钮（Split Button）和一个链接按钮（Link Button）：

```
    <div style="border:1px solid #ccc;background:#fafafa;padding:5p
        <a href="#" class="easyui-splitbutton" menu="#mm" iconCls='
        <a href="#" class="easyui-linkbutton" plain="true" iconCls=
    </div>
    <div id="mm" style="width:150px;">
        <div iconCls="icon-undo">Undo</div>
        <div iconCls="icon-redo">Redo</div>
        <div class="menu-sep"></div>
        <div>Cut</div>
        <div>Copy</div>
        <div>Paste</div>
        <div class="menu-sep"></div>
        <div>
            <span>Open</span>
            <div style="width:150px;">
                <div>Firefox</div>
                <div>Internet Explorer</div>
                <div class="menu-sep"></div>
                <div>Select Program...</div>
            </div>
        </div>
        <div iconCls="icon-remove">Delete</div>
        <div>Select All</div>
    </div>
```

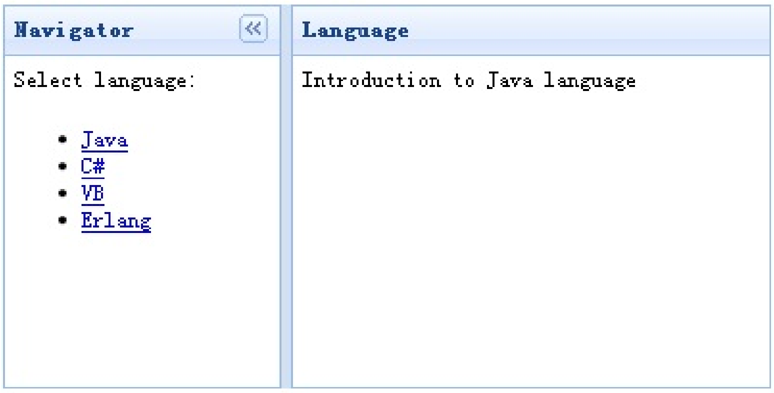
现在已经定义好了一个分割按钮（Split Button），您不需要写任何的 javascript 代码。

jQuery EasyUI 布局

jQuery EasyUI 布局 - 为网页创建边框布局

边框布局（border layout）提供五个区域：east、west、north、south、center。以下是一些通常用法：

- north 区域可以用来显示网站的标语。
- south 区域可以用来显示版权以及一些说明。
- west 区域可以用来显示导航菜单。
- east 区域可以用来显示一些推广的项目。
- center 区域可以用来显示主要的内容。



为了应用布局（layout），您应该确定一个布局（layout）容器，然后定义一些区域。布局（layout）必须至少需要一个 center 区域，以下是一个布局（layout）实例：

```
<div class="easyui-layout" style="width:400px;height:200px;">
    <div region="west" split="true" title="Navigator" style="wi
        <p style="padding:5px;margin:0;">Select language:</p>
        <ul>
            <li><a href="javascript:void(0)" onclick="showconte
            <li><a href="javascript:void(0)" onclick="showconte
            <li><a href="javascript:void(0)" onclick="showconte
            <li><a href="javascript:void(0)" onclick="showconte
        </ul>
    </div>
    <div id="content" region="center" title="Language" style="p
    </div>
</div>
```

我们在一个 <div> 容器中创建了一个边框布局（border layout），布局（layout）把容器切割为两个部分，左边是导航菜单，右边是主要内容。

最后我们写一个 onclick 事件处理函数来检索数据，'showcontent' 函数非常简单：

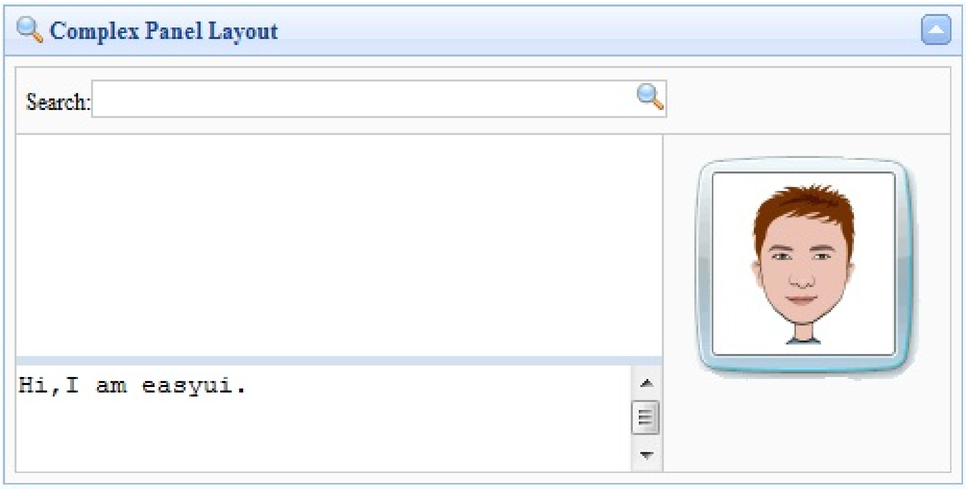
```
function showcontent(language){
    $('#content').html('Introduction to ' + language + ' langua
}
```

下载 jQuery EasyUI 实例

jeasyui-layout-layout.zip

jQuery EasyUI 布局 - 在面板中创建复杂布局

面板（Panel）允许您创建用于多种用途的自定义布局。在本实例中，我们使用面板（panel）和布局（layout）插件来创建一个 msn 消息框。



我们在区域面板中使用多个布局（layout）。在消息框的顶部我们放置一个查询输入框，同时在右边放置一个人物图片。在中间的区域我们通过设置 split 属性为 true，把这部分切割为两部分，允许用户改变区域面板的尺寸大小。

以下就是所有代码：

```
    <div class="easyui-panel" title="Complex Panel Layout" iconCls=
        <div class="easyui-layout" fit="true">
            <div region="north" border="false" class="p-search">
                <label>Search:</label><input></input>
            </div>
            <div region="center" border="false">
                <div class="easyui-layout" fit="true">
                    <div region="east" border="false" class="p-rigl
                        <img src="images/msn.gif"/>
                    </div>
                    <div region="center" border="false" style="borc
                        <div class="easyui-layout" fit="true">
                            <div region="south" split="true" border
                                <textarea style="border:0;width:10(
                            </div>
                            <div region="center" border="false">
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

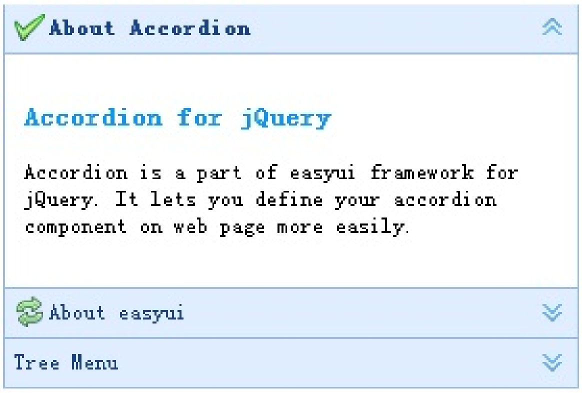
我们不需要写任何的 javascript 代码，它自己有非常强大的设计用户界面的功能。

下载 jQuery EasyUI 实例

jeasyui-layout-panel.zip

jQuery EasyUI 布局 - 创建折叠面板

在本教程中，您将会学习到关于 easyui 折叠面板（Accordion）的知识。 折叠面板（Accordion）包含一系列的面板（panel）。 所有面板（panel）的头部（header）都是可见的，但是一次仅仅显示一个面板（panel）的 body 内容。 当用户点击面板（panel）的头部（header）时，该面板（panel）的 body 内容将可见，同时其他面板（panel）的 body 内容将隐藏不可见。



我们将创建三个面板（panel），第三个面板（panel）包含一个树形菜单。

```
    <div class="easyui-accordion" style="width:300px;height:200px;'
        <div title="About Accordion" iconCls="icon-ok" style="overf
            <h3 style="color:#0099FF;">Accordion for jQuery</h3>
            <p>Accordion is a part of easyui framework for jQuery.
        </div>
        <div title="About easyui" iconCls="icon-reload" selected="t
            easyui help you build your web page easily
        </div>
        <div title="Tree Menu">
            <ul id="tt1" class="easyui-tree">
                <li>
                    <span>Folder1</span>
                    <ul>
                        <li>
                            <span>Sub Folder 1</span>
                            <ul>
                                <li><span>File 11</span></li>
                                <li><span>File 12</span></li>
                                <li><span>File 13</span></li>
                            </ul>
                        </li>
                        <li><span>File 2</span></li>
                        <li><span>File 3</span></li>
                    </ul>
                </li>
                <li><span>File2</span></li>
            </ul>
        </div>
    </div>
```

下载 jQuery EasyUI 实例

jeasyui-layout-accordion.zip

jQuery EasyUI 布局 - 创建标签页（Tabs）

本教程将向您延时如何使用 easyui 创建一个 Tabs 组件。 Tabs 有多个可以动态地添加或移除的面板（panel）。 您可以使用 Tabs 来在相同的页面上显示不同的实体。

Tabs 一次仅仅显示一个面板（panel），每个面板（panel）都有标题、图标和关闭按钮。 当 Tabs 被选中时，将显示对应的面板（panel）的内容。



从 HTML 标记创建 Tabs，包含一个 DIV 容器和一些 DIV 面板（panel）。

```
<div class="easyui-tabs" style="width:400px;height:100px;">
    <div title="First Tab" style="padding:10px;">
        First Tab
    </div>
    <div title="Second Tab" closable="true" style="padding:10p
        Second Tab
    </div>
    <div title="Third Tab" iconCls="icon-reload" closable="true
        Third Tab
    </div>
</div>
```

我们创建一个带有三个面板（panel）的 Tabs 组件，第二个和第三个面板（panel）可以通过点击关闭按钮进行关闭。

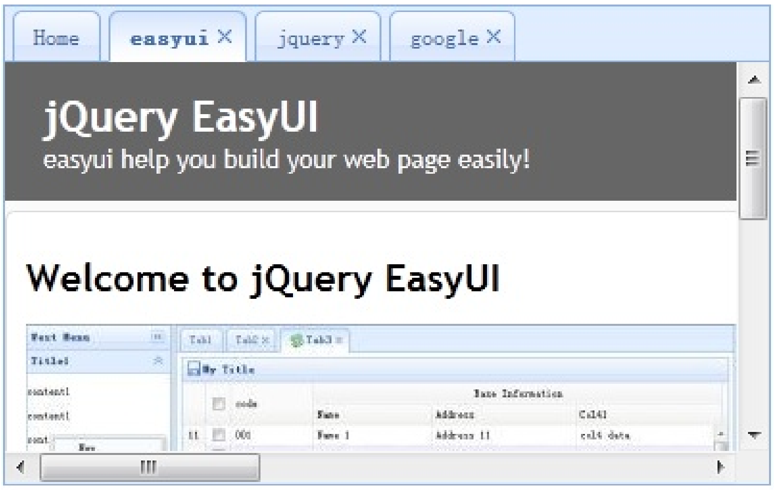
下载 jQuery EasyUI 实例

jeasyui-layout-tabs1.zip

jQuery EasyUI 布局 - 动态添加标签页（Tabs）

通过使用 jQuery EasyUI 可以很容易地添加 Tabs。您只需要调用 'add' 方法即可。

在本教程中，我们将使用 iframe 动态地添加显示在一个页面上的 Tabs。 当点击添加按钮，一个新的 tab 将被添加。如果 tab 已经存在，它将被激活。



步骤 1：创建 Tabs

```
<div style="margin-bottom:10px">
    <a href="#" class="easyui-linkbutton" onclick="addTab('goog
    <a href="#" class="easyui-linkbutton" onclick="addTab('jque
    <a href="#" class="easyui-linkbutton" onclick="addTab('easy
</div>
<div id="tt" class="easyui-tabs" style="width:400px;height:250p
    <div title="Home">
    </div>
</div>
```

这个 html 代码非常简单,我们创建了带有一个名为 'Home' 的 tab 面板的 Tabs。请注意，我们不需要写任何的 js 代码。

步骤 2：实现 'addTab' 函数

```
    function addTab(title, url){
        if ($('#tt').tabs('exists', title)){
            $('#tt').tabs('select', title);
        } else {
            var content = '<iframe scrolling="auto" frameborder="0'
            $('#tt').tabs('add',{
                title:title,
                content:content,
                closable:true
            });
        }
    }
```

我们使用 'exists' 方法来判断 tab 是否已经存在，如果已存在则激活 tab。如果不存在则调用 'add' 方法来添加一个新的 tab 面板。

下载 jQuery EasyUI 实例

jeasyui-layout-tabs2.zip

jQuery EasyUI 布局 - 添加自动播放标签页（Tabs）

本教程将向您展示如何创建一个自动播放的 Tabs。 Tabs 组件显示第一个 tab 面板，然后显示第二个、第三个... 我们将写一些代码来自动地切换 tab 面板，然后让它循环。



步骤 1：创建 Tabs

```
<div id="tt" class="easyui-tabs" style="width:330px;height:270p
    <div title="Shirt1" style="padding:20px;">
        <img src="images/shirt1.gif">
    </div>
    <div title="Shirt2" style="padding:20px;">
        <img src="images/shirt2.gif">
    </div>
    <div title="Shirt3" style="padding:20px;">
        <img src="images/shirt3.gif">
    </div>
    <div title="Shirt4" style="padding:20px;">
        <img src="images/shirt4.gif">
    </div>
    <div title="Shirt5" style="padding:20px;">
        <img src="images/shirt5.gif">
    </div>
</div>
```

步骤 2：写自动播放代码

```
    var index = 0;
    var t = $('#tt');
    var tabs = t.tabs('tabs');
    setInterval(function(){
        t.tabs('select', tabs[index].panel('options').title);
        index++;
        if (index >= tabs.length){
            index = 0;
        }
    }, 3000);
```

正如您所看到的，我们调用 'tabs' 方法来得到所有 tab 面板，并使用 'setInterval' 函数来切换他们。

下载 jQuery EasyUI 实例

jeasyui-layout-tabs3.zip

jQuery EasyUI 布局 - 创建 XP 风格左侧面板

通常情况下，在 Windows XP 的资源管理器文件夹中，左侧的面板（panel）包含一些常见任务。 本教程向您展示如何通过 easyui 的面板（panel）插件来创建 XP 左侧面板。



定义一些面板（panel）

我们定义一些面板（panel），这些面板（panel）用来显示一些任务。每个面板（panel）应该至少有折叠/展开工具按钮。

代码如下所示：

```
<div style="width:200px;height:auto;background:#7190E0;padding
    <div class="easyui-panel" title="Picture Tasks" collapsible
        View as a slide show<br/>
        Order prints online<br/>
        Print pictures
    </div>
    <br/>
    <div class="easyui-panel" title="File and Folder Tasks" co
        Make a new folder<br/>
        Publish this folder to the Web<br/>
        Share this folder
    </div>
    <br/>
    <div class="easyui-panel" title="Other Places" collapsible=
        New York<br/>
        My Pictures<br/>
        My Computer<br/>
        My Network Places
    </div>
    <br/>
    <div class="easyui-panel" title="Details" collapsible="true
        My documents<br/>
        File folder<br/><br/>
        Date modified: Oct.3rd 2010
    </div>
</div>
```

自定义面板（panel）的外观效果

请注意，这个视图外观效果不是我们想要的，我们必须改变面板（panel）的头部背景图片和折叠/展开按钮的图标。

做到这一点并不难，我们需要做的只是重新定义一些 CSS。

```
.panel-body{
    background:#f0f0f0;
}
.panel-header{
    background:#fff url('images/panel_header_bg.gif') no-repeat
}
.panel-tool-collapse{
    background:url('images/arrow_up.gif') no-repeat 0px -3px;
}
.panel-tool-expand{
    background:url('images/arrow_down.gif') no-repeat 0px -3px
}
```

由此可见，使用 easyui 定义用户界面非常简单。

下载 jQuery EasyUI 实例

jeasyui-layout-xp.zip

jQuery EasyUI 数据网格

jQuery EasyUI 数据网格 - 转换 HTML 表格为数据网格

本实例演示如何转换表格（table）为数据网格（datagrid）。

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

数据网格（datagrid）的列信息是定义在 <thead> 标记中，数据是定义在 <tbody> 标记中。确保为所有的数据列设置 field 名称，请看下面的实例：

```
<table id="tt" class="easyui-datagrid" style="width:400px;heigh
    <thead>
        <tr>
            <th field="name1" width="50">Col 1</th>
            <th field="name2" width="50">Col 2</th>
            <th field="name3" width="50">Col 3</th>
            <th field="name4" width="50">Col 4</th>
            <th field="name5" width="50">Col 5</th>
            <th field="name6" width="50">Col 6</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
            <td>Data 4</td>
            <td>Data 5</td>
            <td>Data 6</td>
        </tr>
    </tbody>
</table>
```

非常棒，您可以定义一个复杂的表头，例如：

```
    <thead>
        <tr>
            <th field="name1" width="50" rowspan="2">Col 1</th>
            <th field="name2" width="50" rowspan="2">Col 2</th>
            <th field="name3" width="50" rowspan="2">Col 3</th>
            <th colspan="3">Details</th>
        </tr>
        <tr>
            <th field="name4" width="50">Col 4</th>
            <th field="name5" width="50">Col 5</th>
            <th field="name6" width="50">Col 6</th>
        </tr>
    </thead>
```

现在您可以看见，复杂表头已经创建。

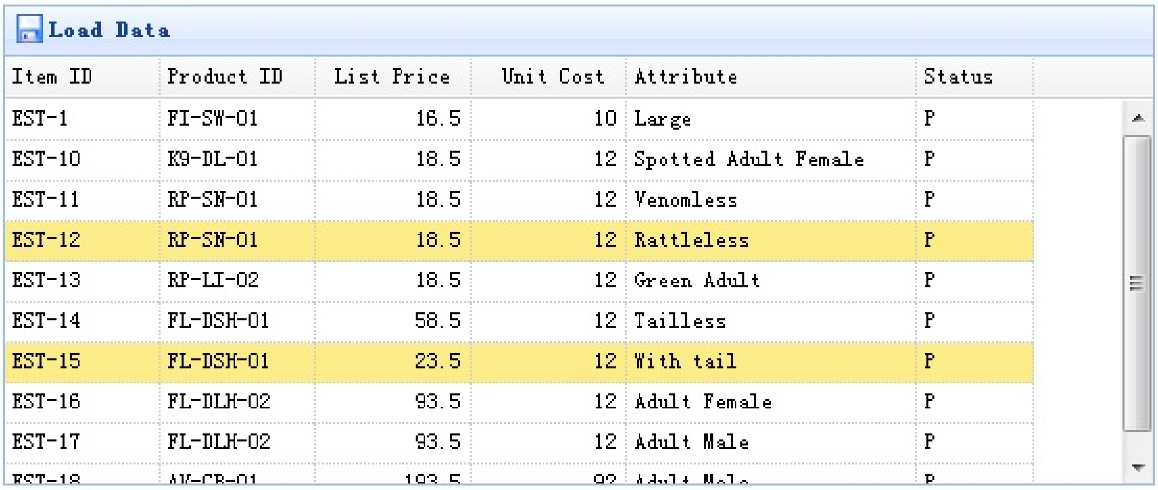
Col 1	Col 2	Col 3	Details		
			Col 4	Col 5	Col 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid1.zip

jQuery EasyUI 数据网格 - 取得选中行数据

本实例演示如何取得选中行数据。



Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

数据网格（datagrid）组件包含两种方法来检索选中行数据：

- getSelected：取得第一个选中行数据，如果没有选中行，则返回 null，否则返回记录。
- getSelections：取得所有选中行数据，返回元素记录的数组数据。

创建数据网格（DataGrid）

```
	<table id="tt" class="easyui-datagrid" style="width:600px;heigh
			url="data/datagrid_data.json"
			title="Load Data" iconCls="icon-save">
		<thead>
			<tr>
				<th field="itemid" width="80">Item ID</th>
				<th field="productid" width="80">Product ID</th>
				<th field="listprice" width="80" align="right">List
				<th field="unitcost" width="80" align="right">Unit
				<th field="attr1" width="150">Attribute</th>
				<th field="status" width="60" align="center">Stauts
			</tr>
		</thead>
	</table>
```

使用演示

取得选中行数据：

```
var row = $('#tt').datagrid('getSelected');
if (row){
    alert('Item ID:'+row.itemid+"\nPrice:"+row.listprice);
}
```

取得所有选中行的 itemid：

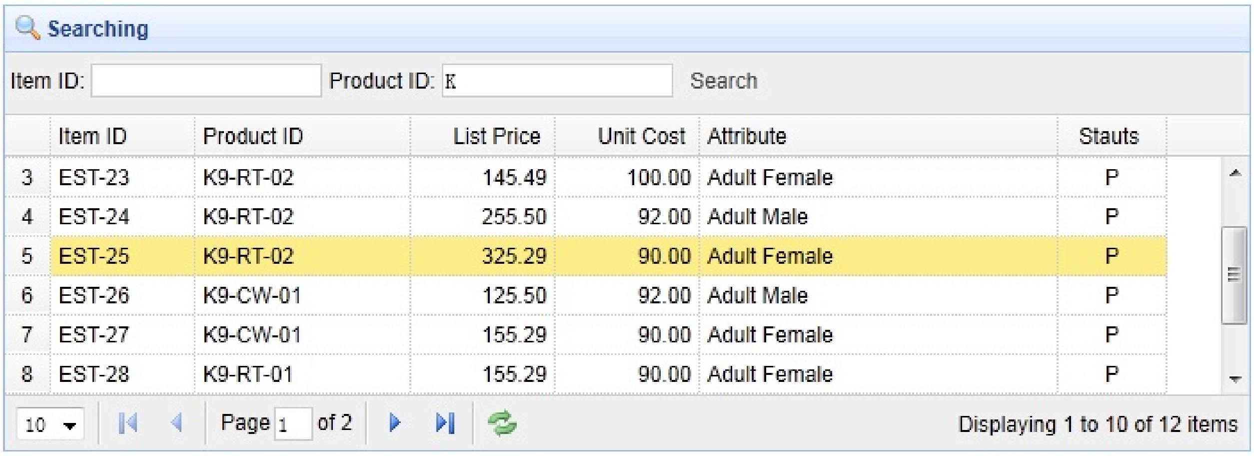
```
var ids = [];
var rows = $('#tt').datagrid('getSelections');
for(var i=0; i<rows.length; i++){
    ids.push(rows[i].itemid);
}
alert(ids.join('\n'));
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid3.zip

jQuery EasyUI 数据网格 - 添加查询功能

本实例演示如何从数据库得到数据，并将它们显示在数据网格（datagrid）中。然后演示如何根据用户输入的搜索关键词搜寻显示结果。



创建数据网格（DataGrid）

创建带有分页功能的数据网格（datagrid），然后添加工具栏到其中。

```
<table id="tt" class="easyui-datagrid" style="width:600px;heigh
        url="datagrid24_getdata.php" toolbar="#tb"
        title="Load Data" iconCls="icon-save"
        rownumbers="true" pagination="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

工具栏定义如下：

```
<div id="tb" style="padding:3px">
    <span>Item ID:</span>
    <input id="itemid" style="line-height:26px;border:1px solid
    <span>Product ID:</span>
    <input id="productid" style="line-height:26px;border:1px so
    <a href="#" class="easyui-linkbutton" plain="true" onclick=
</div>
```

当用户输入查询值并按下查询按钮时，'doSearch' 函数将被调用：

```
function doSearch(){
    $('#tt').datagrid('load',{
        itemid: $('#itemid').val(),
        productid: $('#productid').val()
    });
}
```

上面的代码调用了 'load' 方法来加载新的数据网格（datagrid）数据。我们需要传递 'itemid' 和 'productid' 参数到服务器。

服务器端代码

```
include 'conn.php';

$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
$itemid = isset($_POST['itemid']) ? mysql_real_escape_string($_
$productid = isset($_POST['productid']) ? mysql_real_escape_str

$offset = ($page-1)*$rows;

$result = array();

$where = "itemid like '$itemid%' and productid like '$productic
$rs = mysql_query("select count(*) from item where " . $where);
$row = mysql_fetch_row($rs);
$result["total"] = $row[0];

$rs = mysql_query("select * from item where " . $where . " lim:

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
$result["rows"] = $items;

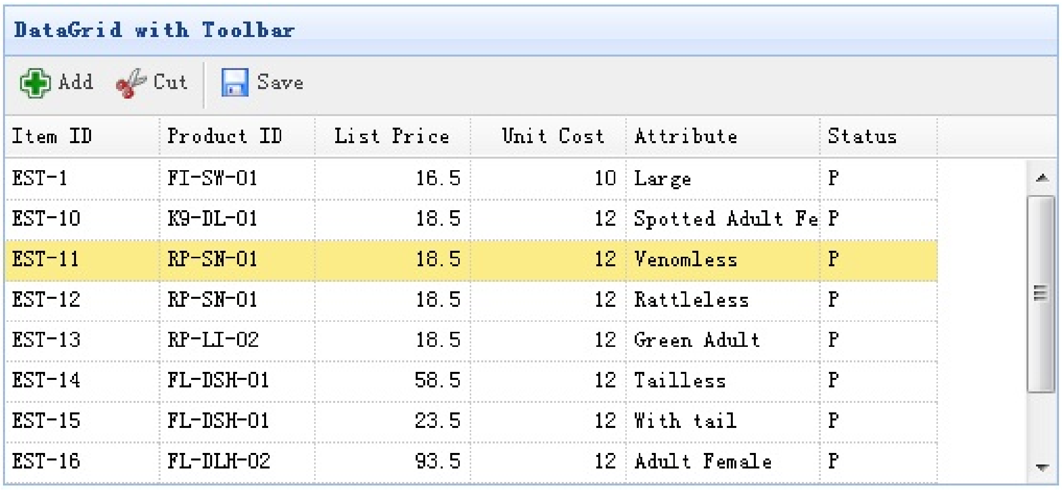
echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid24.zip

jQuery EasyUI 数据网格 - 添加工具栏

本实例演示如何添加工具栏（toolbar）到数据网格（datagrid）。



创建数据网格（DataGrid）

```
<table id="tt" class="easyui-datagrid" style="width:600px;heigh
        url="data/datagrid_data.json"
        title="DataGrid with Toolbar" iconCls="icon-save"
        toolbar="#tb">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
<div id="tb">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" pl
    <a href="#" class="easyui-linkbutton" iconCls="icon-cut" pl
    <a href="#" class="easyui-linkbutton" iconCls="icon-save" p
</div>
```

我们不需要写任何的 javascript 代码，只需通过 'toolbar' 属性附加工具栏（toolbar）到数据网格（datagrid）。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid4.zip

jQuery EasyUI 数据网格 - 创建复杂工具栏

数据网格（datagrid）的工具栏（toolbar）可以包含按钮及其他组件。 您可以通个一个已存在的 DIV 标签来简单地定义工具栏布局，该 DIV 标签将成为数据网格（datagrid）工具栏的内容。 本教程将向您展示如何创建数据网格（datagrid）组件的复杂工具栏。

创建工具栏（Toolbar）

```
<div id="tb" style="padding:5px;height:auto">
    <div style="margin-bottom:5px">
        <a href="#" class="easyui-linkbutton" iconCls="icon-add
        <a href="#" class="easyui-linkbutton" iconCls="icon-edi
        <a href="#" class="easyui-linkbutton" iconCls="icon-sav
        <a href="#" class="easyui-linkbutton" iconCls="icon-cut
        <a href="#" class="easyui-linkbutton" iconCls="icon-rem
    </div>
    <div>
        Date From: <input class="easyui-datebox" style="width:8
        To: <input class="easyui-datebox" style="width:80px">
        Language:
        <input class="easyui-combobox" style="width:100px"
                url="data/combobox_data.json"
                valueField="id" textField="text">
        <a href="#" class="easyui-linkbutton" iconCls="icon-sea
    </div>
</div>
```

创建数据网格（DataGrid）

```
<table class="easyui-datagrid" style="width:600px;height:250px'
        url="data/datagrid_data.json"
        title="DataGrid - Complex Toolbar" toolbar="#tb"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="60">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" align="right" width="70">List
            <th field="unitcost" align="right" width="70">Unit
            <th field="attr1" width="200">Address</th>
            <th field="status" width="50">Status</th>
        </tr>
    </thead>
</table>
```

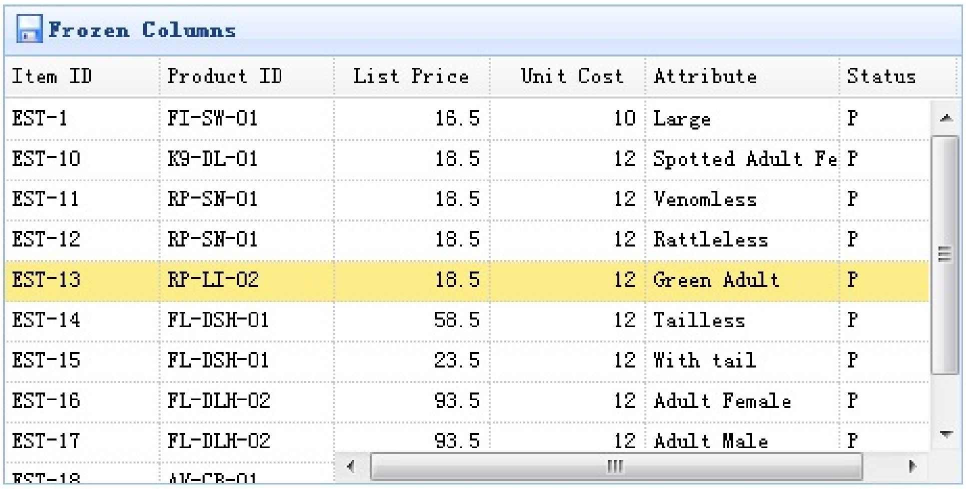
正如您所看到的，数据网格（datagrid）的工具栏域对话框（dialog）相似。我们不需要写任何的 javascript 代码，就能创建带有复杂工具栏的数据网格（datagrid）。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid19.zip

jQuery EasyUI 数据网格 - 设置冻结列

本实例演示如何冻结一些列，当用户在网格上移动水平滚动条时，冻结列不能滚动到视图的外部。



Frozen Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fe	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

为了冻结列，您需要定义 frozenColumns 属性。frozenColumn 属性和 columns 属性一样。

```
$('#tt').datagrid({
    title:'Frozen Columns',
    iconCls:'icon-save',
    width:500,
    height:250,
    url:'data/datagrid_data.json',
    frozenColumns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
    ]],
    columns:[[
        {field:'listprice',title:'List Price',width:80,align:'r
        {field:'unitcost',title:'Unit Cost',width:80,align:'ri
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

您不需要写任何的 javascript 代码，这样您就能创建一个数据网格（datagrid）组件，如下所示：

```
<table id="tt" title="Frozen Columns" class="easyui-datagrid" s
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save">
    <thead frozen="true">
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
        </tr>
    </thead>
    <thead>
        <tr>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid5.zip

jQuery EasyUI 数据网格 - 动态改变列

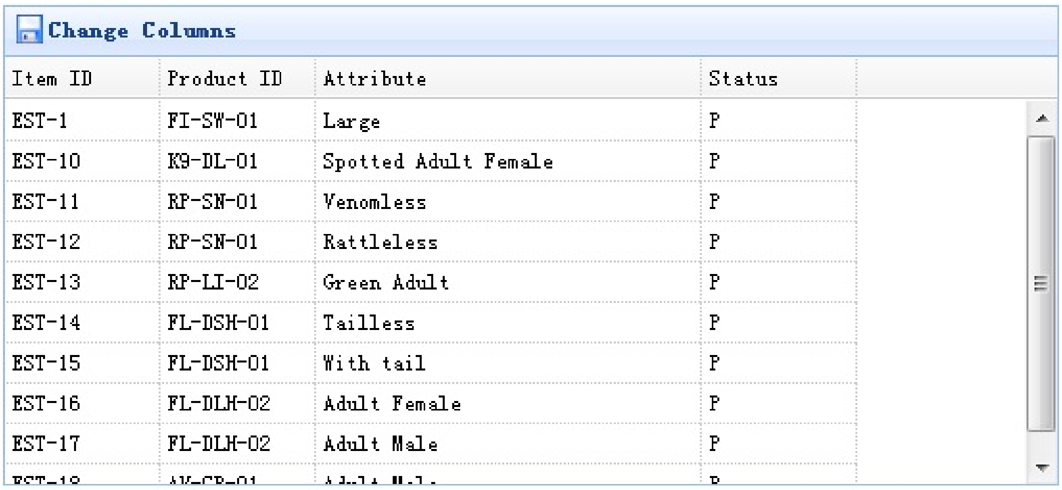
数据网格（DataGrid）列可以使用 'columns' 属性简单地定义。如果您想动态地改变列，那根本没有问题。为了改变列，您可以重新调用datagrid 方法，并传递一个新的 columns 属性。

创建数据网格（DataGrid）

```
<table id="tt" title="Frozen Columns" class="easyui-datagrid" s
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save">
</table>
```

```
$('#tt').datagrid({
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'attr1',title:'Attribute',width:200},
        {field:'status',title:'Status',width:80}
    ]]
});
```

运行网页，您将看见：

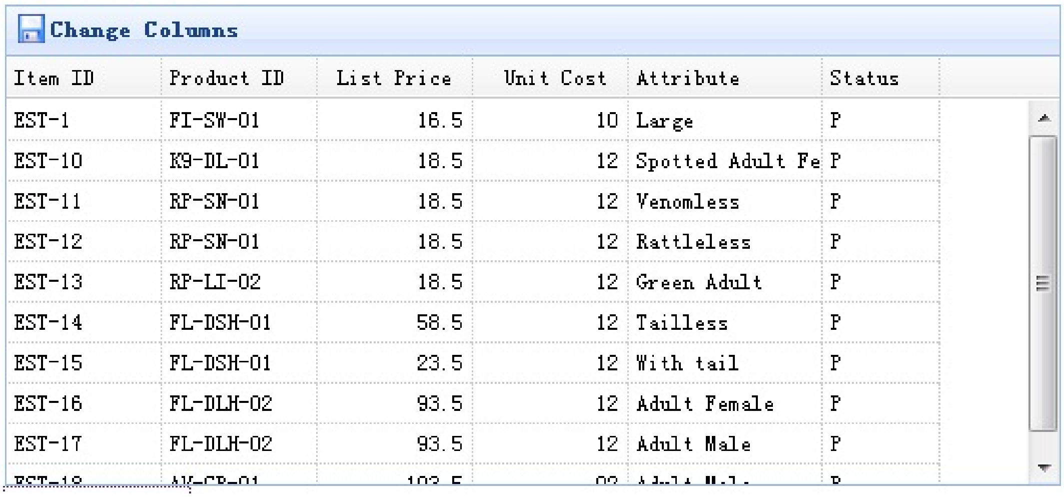


Item ID	Product ID	Attribute	Status
EST-1	FI-SW-01	Large	P
EST-10	K9-DL-01	Spotted Adult Female	P
EST-11	RP-SN-01	Venomless	P
EST-12	RP-SN-01	Rattleless	P
EST-13	RP-LI-02	Green Adult	P
EST-14	FL-DSH-01	Tailless	P
EST-15	FL-DSH-01	With tail	P
EST-16	FL-DLH-02	Adult Female	P
EST-17	FL-DLH-02	Adult Male	P

可是有时候您想改变列，所以您需要写一些代码：

```
$('#tt').datagrid({
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List Price',width:80,align:'r
        {field:'unitcost',title:'Unit Cost',width:80,align:'rig
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

请记住，我们已经定义了其他属性，比如：url、width、height 等等。我们不需要再一次定义它们，我们定义那些我们需要改变的。



Change Columns

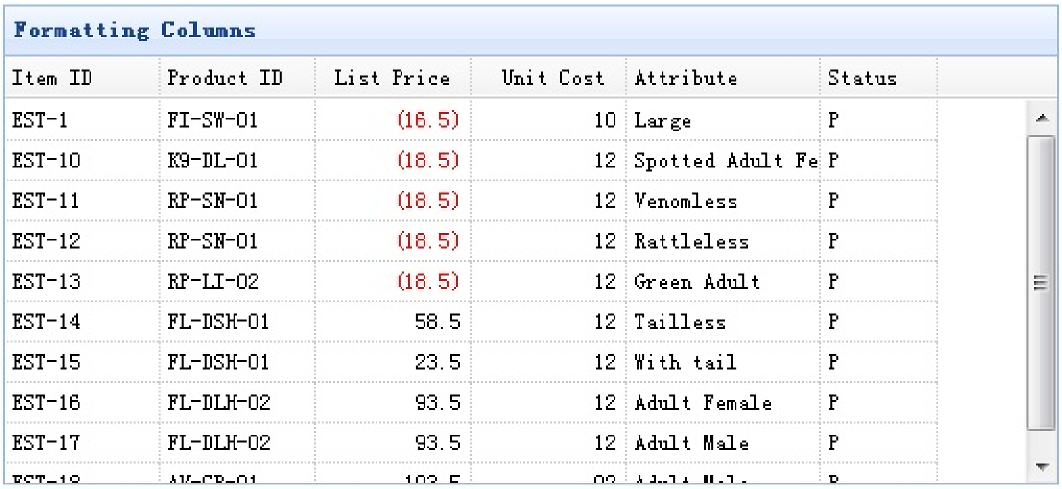
Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fe	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid6.zip

jQuery EasyUI 数据网格 - 格式化列

以下实例格式化在 easyui DataGrid 里的列数据，并使用自定义列的 formatter，如果价格小于 20 就将文本变为红色。



Formatting Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	(16.5)	10	Large	P
EST-10	K9-DL-01	(18.5)	12	Spotted Adult Fe	P
EST-11	RP-SN-01	(18.5)	12	Venomless	P
EST-12	RP-SN-01	(18.5)	12	Rattleless	P
EST-13	RP-LI-02	(18.5)	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

为了格式化一个数据网格（DataGrid）列，我们需要设置 formatter 属性，它是一个函数。这个格式化函数包含三个参数：

- value：当前列对应字段值。
- row：当前的行记录数据。
- index：当前的行下标。

创建数据网格（DataGrid）

```
    <table id="tt" title="Formatting Columns" class="easyui-datagri
            url="data/datagrid_data.json"
            singleSelect="true" iconCls="icon-save">
        <thead>
            <tr>
                <th field="itemid" width="80">Item ID</th>
                <th field="productid" width="80">Product ID</th>
                <th field="listprice" width="80" align="right" forr
                <th field="unitcost" width="80" align="right">Unit 
                <th field="attr1" width="100">Attribute</th>
                <th field="status" width="60" align="center">Stauts
            </tr>
        </thead>
    </table>
```

请注意，'listprice' 字段有一个 'formatter' 属性，用来指明格式化函数。

写格式化函数

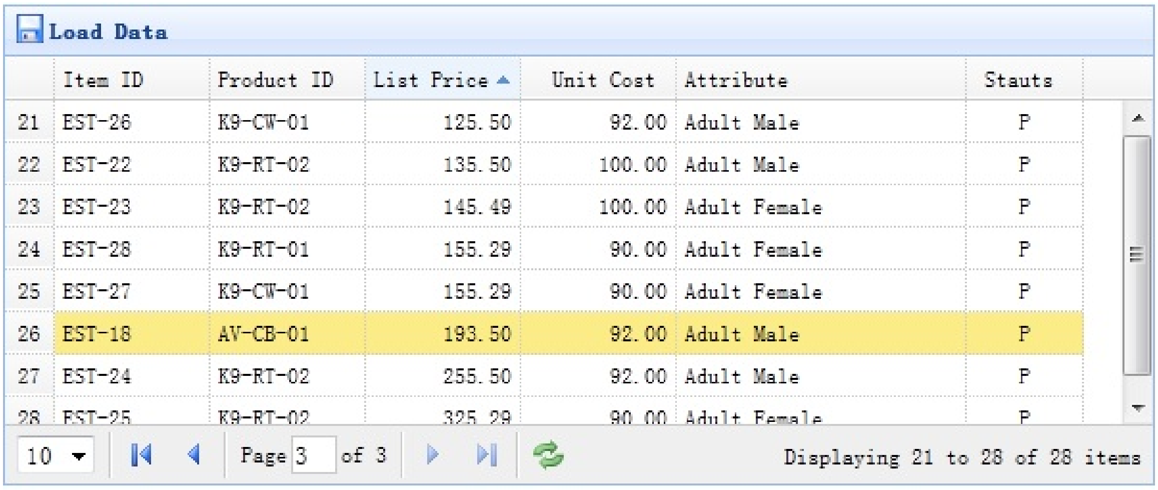
```
function formatPrice(val,row){
    if (val < 20){
        return '<span style="color:red;">('+val+')</span>';
    } else {
        return val;
    }
}
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid7.zip

jQuery EasyUI 数据网格 - 设置排序

本实例演示如何通过点击列表头来排序数据网格（DataGrid）。



数据网格（DataGrid）的所有列可以通过点击列表头来排序。您可以定义哪列可以排序。默认的，列是不能排序的，除非您设置 sortable 属性为 true。

创建数据网格（DataGrid）

```
    <table id="tt" class="easyui-datagrid" style="width:600px;heigh
            url="datagrid8_getdata.php"
            title="Load Data" iconCls="icon-save"
            rownumbers="true" pagination="true">
        <thead>
            <tr>
                <th field="itemid" width="80" sortable="true">Item
                <th field="productid" width="80" sortable="true">Pr
                <th field="listprice" width="80" align="right" sort
                <th field="unitcost" width="80" align="right" sorta
                <th field="attr1" width="150">Attribute</th>
                <th field="status" width="60" align="center">Stauts
            </tr>
        </thead>
    </table>
```

我们定义一些可排序的列，包含 itemid、productid、listprice、unitcost 等等。'attr1' 列和 'status' 列不能排序。

当排序时，数据网格（DataGrid）将发送两个参数到远程服务器：

- sort：排序列字段名。
- order：排序方式，可以是 'asc' 或者 'desc'，默认值是 'asc'。

服务器端代码

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
$sort = isset($_POST['sort']) ? strval($_POST['sort']) : 'item
$order = isset($_POST['order']) ? strval($_POST['order']) : 'as
$offset = ($page-1)*$rows;

$result = array();

include 'conn.php';

$rs = mysql_query("select count(*) from item");
$row = mysql_fetch_row($rs);
$result["total"] = $row[0];

$rs = mysql_query("select * from item order by $sort $order lir

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
$result["rows"] = $items;

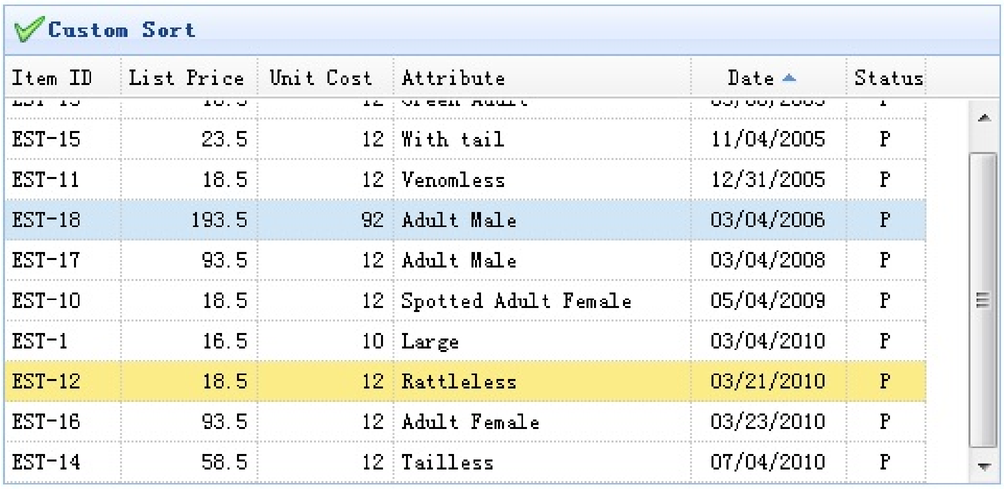
echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid8.zip

jQuery EasyUI 数据网格 - 自定义排序

如果默认的排序行为不满足您的需求，您可以自定义数据网格（datagrid）的排序行为。



Custom Sort

Item ID	List Price	Unit Cost	Attribute	Date	Status
EST-15	23.5	12	With tail	11/04/2005	P
EST-11	18.5	12	Venomless	12/31/2005	P
EST-18	193.5	92	Adult Male	03/04/2006	P
EST-17	93.5	12	Adult Male	03/04/2008	P
EST-10	18.5	12	Spotted Adult Female	05/04/2009	P
EST-1	16.5	10	Large	03/04/2010	P
EST-12	18.5	12	Rattleless	03/21/2010	P
EST-16	93.5	12	Adult Female	03/23/2010	P
EST-14	58.5	12	Tailless	07/04/2010	P

最基础的，用户可以在列上定义一个排序函数，函数名是 sorter。这个函数将接受两个值，返回值将如下：

valueA > valueB => 返回 1

valueA < valueB => 返回 -1

自定义排序代码

```
<table id="tt"></table>
```

```
$('#tt').datagrid({
    title:'Custom Sort',
    iconCls:'icon-ok',
    width:520,
    height:250,
    singleSelect:true,
    remoteSort:false,
    columns:[[
        {field:'itemid',title:'Item ID',width:60,sortable:true]
        {field:'listprice',title:'List Price',width:70,align:'r
        {field:'unitcost',title:'Unit Cost',width:70,align:'rig
        {field:'attr1',title:'Attribute',width:120,sortable:tru
        {field:'date',title:'Date',width:80,sortable:true,align
            sorter:function(a,b){
                a = a.split('/');
                b = b.split('/');
                if (a[2] == b[2]){
                    if (a[0] == b[0]){
                        return (a[1]>b[1]?1:-1);
                    } else {
                        return (a[0]>b[0]?1:-1);
                    }
                } else {
                    return (a[2]>b[2]?1:-1);
                }
            }
        },
        {field:'status',title:'Status',width:40,align:'center']
    ]]
}).datagrid('loadData', data);
```

您可以从这段代码中看到，我们为 date 列创建了自定义的 sorter。日期的格式是 'dd/mm/yyyy'，可以轻松的按年月日排序。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid14.zip

jQuery EasyUI 数据网格 - 创建列组合

easyui 的数据网格（DataGrid）可以创建列组合，如下所示：

Column Group

	Item ID	Product ID	Item Details			
			List Price	Unit Cost	Attribute	Status
1	EST-1	FI-SW-01	16.5	10	Large	P
2	EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
3	EST-11	RP-SN-01	18.5	12	Venomless	P
4	EST-12	RP-SN-01	18.5	12	Rattleless	P
5	EST-13	RP-LI-02	18.5	12	Green Adult	P
6	EST-14	FL-DSH-01	58.5	12	Tailless	P
7	EST-15	FL-DSH-01	23.5	12	With tail	P
8	EST-16	FL-DLH-02	93.5	12	Adult Female	P

在本实例中，我们使用平面数据来填充数据网格（DataGrid）的数据，并把 listprice、unitcost、addr1、status 列组合在一个单一的列下。

为了创建列组合，您应该定义数据网格（datagrid）插件的 columns 数据。列的每个元素是定义一组可使用 rowspan 或 colspan 属性来进行组合的单元格。

下面的代码实现了上面的实例：

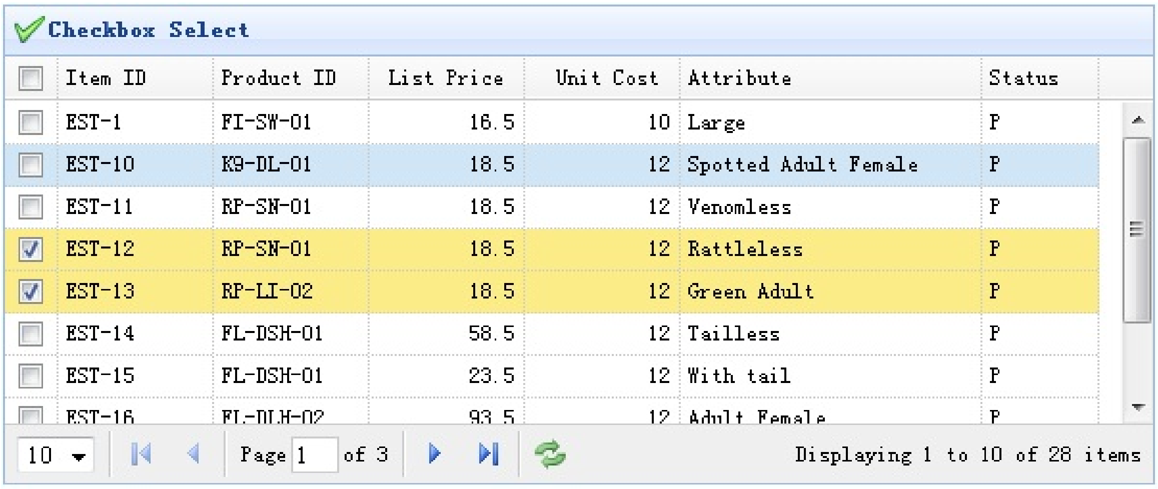
```
<table id="tt" title="Column Group" class="easyui-datagrid" sty
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save" rownumbers="tru
    <thead>
        <tr>
            <th rowspan="2" field="itemid" width="80">Item ID</
            <th rowspan="2" field="productid" width="80">Produc
            <th colspan="4">Item Details</th>
        </tr>
        <tr>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit 
            <th field="attr1" width="100">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid9.zip

jQuery EasyUI 数据网格 - 添加复选框

本实例演示如何放置一个复选框列到数据网格（DataGrid）。通过复选框，用户将可以选择 选中/取消选中 网格行数据。



为了添加一个复选框列，我们仅仅需要添加一个列的 checkbox 属性，并设置它为 true。代码如下所示：

```
<table id="tt" title="Checkbox Select" class="easyui-datagrid"
        url="data/datagrid_data.json"
        idField="itemid" pagination="true"
        iconCls="icon-save">
    <thead>
        <tr>
            <th field="ck" checkbox="true"></th>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="100">Attribute</th>
            <th field="status" width="60" align="center">Status
        </tr>
    </thead>
</table>
```

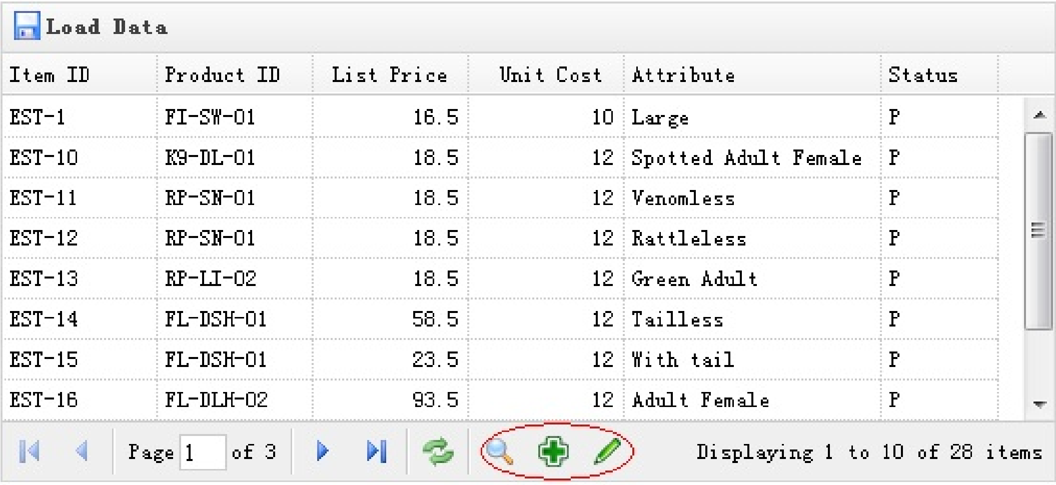
以上代码添加了一个带有 checkbox 属性的列，所以它将成为复选框列。如果 idField 属性已设置，数据网格（DataGrid）的选择集合将在不同的页面保持。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid10.zip

jQuery EasyUI 数据网格 - 自定义分页

数据网格（datagrid）内置一个很好特性的分页功能，自定义也相当简单。在本教程中，我们将创建一个数据网格（datagrid），并在分页工具栏上添加一些自定义按钮。



Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P

Page 1 of 3 Displaying 1 to 10 of 28 items

创建数据网格（DataGrid）

```
    <table id="tt" title="Load Data" class="easyui-datagrid" style=
            url="data/datagrid_data.json"
            iconCls="icon-save" pagination="true">
        <thead>
            <tr>
                <th field="itemid" width="80">Item ID</th>
                <th field="productid" width="80">Product ID</th>
                <th field="listprice" width="80" align="right">List
                <th field="unitcost" width="80" align="right">Unit
                <th field="attr1" width="100">Attribute</th>
                <th field="status" width="60" align="center">Stauts
            </tr>
        </thead>
    </table>
```

请记得，设置 'pagination' 属性为 true，这样才会生成分页工具栏。

自定义分页工具栏

```
    var pager = $('#tt').datagrid('getPager');    // get the pager
    pager.pagination({
        showPageList:false,
        buttons:[{
            iconCls:'icon-search',
            handler:function(){
                alert('search');
            }
        },{
            iconCls:'icon-add',
            handler:function(){
                alert('add');
            }
        },{
            iconCls:'icon-edit',
            handler:function(){
                alert('edit');
            }
        }],
        onBeforeRefresh:function(){
            alert('before refresh');
            return true;
        }
    });
```

正如您所看到的，我们首先得到数据网格（datagrid）的 pager 对象，然后重新创建分页（pagination）。我们隐藏页面列表，并添加三个新的按钮。

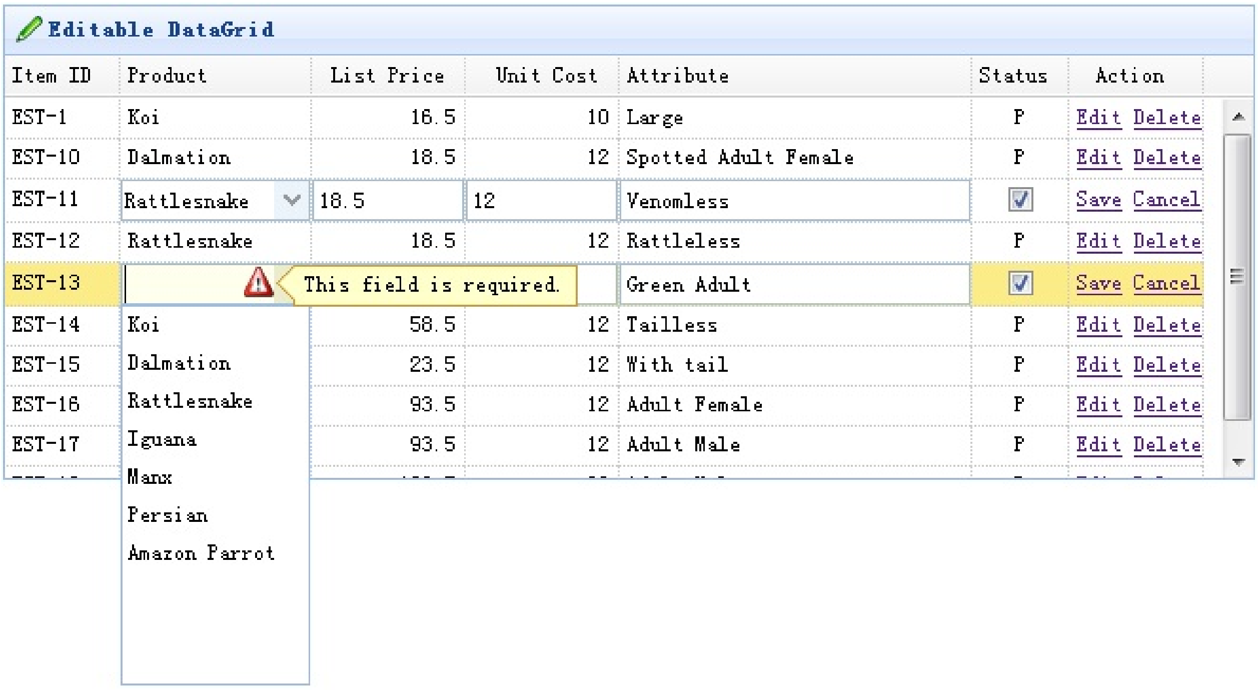
下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid11.zip

jQuery EasyUI 数据网格 - 启用行内编辑

可编辑的功能是最近添加到数据网格（datagrid）的。它可以使用户添加一个新行到数据网格（datagrid）。用户也可以更新一个或多个行。

本教程向您展示如何创建一个数据网格（datagrid）和内联编辑器。



创建数据网格（DataGrid）

```
$(function(){
    $('#tt').datagrid({
        title:'Editable DataGrid',
        iconCls:'icon-edit',
        width:660,
        height:250,
        singleSelect:true,
        idField:'itemid',
        url:'datagrid_data.json',
        columns:[[
            {field:'itemid',title:'Item ID',width:60},
            {field:'productid',title:'Product',width:100,
                formatter:function(value){
                    for(var i=0; i<products.length; i++){
                        if (products[i].productid == value) ret
                    }
                    return value;
                },
                editor:{
                    type:'combobox',
```

```
                        options:{
                            valueField:'productid',
                            textField:'name',
                            data:products,
                            required:true
                        }
                    }
                },
                {field:'listprice',title:'List Price',width:80,alig
                {field:'unitcost',title:'Unit Cost',width:80,align:
                {field:'attr1',title:'Attribute',width:150,editor:
                {field:'status',title:'Status',width:50,align:'cent
                    editor:{
                        type:'checkbox',
                        options:{
                            on: 'P',
                            off: ''
                        }
                    }
                },
                {field:'action',title:'Action',width:70,align:'cent
                    formatter:function(value,row,index){
                        if (row.editing){
                            var s = '<a href="#" onclick="saverow(t
                            var c = '<a href="#" onclick="cancelrow
                            return s+c;
                        } else {
                            var e = '<a href="#" onclick="editrow(t
                            var d = '<a href="#" onclick="deleterow
                            return e+d;
                        }
                    }
                }
            ]],
            onBeforeEdit:function(index,row){
                row.editing = true;
                updateActions(index);
            },
            onAfterEdit:function(index,row){
                row.editing = false;
                updateActions(index);
            },
            onCancelEdit:function(index,row){
                row.editing = false;
                updateActions(index);
            }
        });
    });
    function updateActions(index){
        $('#tt').datagrid('updateRow',{
            index: index,
            row:{}
        });
```

```
    }
```

为了启用数据网格行内编辑，您应该添加一个 editor 属性到列中。编辑器（editor）会告诉数据网格（datagrid）如何编辑字段及如何保存字段值。正如您所看到的，我们定义的三个编辑器（editor）：text、combobox 和 checkbox。

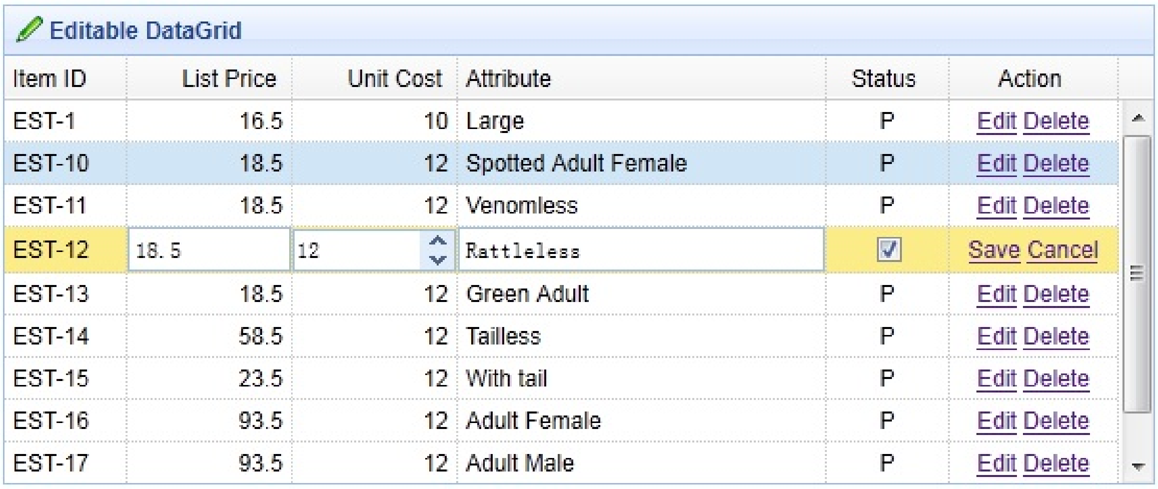
```
    function getRowIndex(target){
        var tr = $(target).closest('tr.datagrid-row');
        return parseInt(tr.attr('datagrid-row-index'));
    }
    function editrow(target){
        $('#tt').datagrid('beginEdit', getRowIndex(target));
    }
    function deleterow(target){
        $.messager.confirm('Confirm','Are you sure?',function(r){
            if (r){
                $('#tt').datagrid('deleteRow', getRowIndex(target)
            }
        });
    }
    function saverow(target){
        $('#tt').datagrid('endEdit', getRowIndex(target));
    }
    function cancelrow(target){
        $('#tt').datagrid('cancelEdit', getRowIndex(target));
    }
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid12.zip

jQuery EasyUI 数据网格 - 扩展编辑器

一些常见的编辑器（editor）添加到数据网格（datagrid），以便用户编辑数据。 所有的编辑器（editor）都定义在 $.fn.datagrid.defaults.editors 对象中，这个可以继承扩展以便支持新的编辑器（editor）。 本教程将向您展示如何添加一个新的 numberspinner 编辑器到数据网格（datagrid）。



Editable DataGrid

Item ID	List Price	Unit Cost	Attribute	Status	Action
EST-1	16.5	10	Large	P	Edit Delete
EST-10	18.5	12	Spotted Adult Female	P	Edit Delete
EST-11	18.5	12	Venomless	P	Edit Delete
EST-12	18.5	12	Rattleless	☑	Save Cancel
EST-13	18.5	12	Green Adult	P	Edit Delete
EST-14	58.5	12	Tailless	P	Edit Delete
EST-15	23.5	12	With tail	P	Edit Delete
EST-16	93.5	12	Adult Female	P	Edit Delete
EST-17	93.5	12	Adult Male	P	Edit Delete

继承扩展 numberspinner 编辑器

```
$.extend($.fn.datagrid.defaults.editors, {
    numberspinner: {
        init: function(container, options){
            var input = $('<input type="text">').appendTo(conta
            return input.numberspinner(options);
        },
        destroy: function(target){
            $(target).numberspinner('destroy');
        },
        getValue: function(target){
            return $(target).numberspinner('getValue');
        },
        setValue: function(target, value){
            $(target).numberspinner('setValue',value);
        },
        resize: function(target, width){
            $(target).numberspinner('resize',width);
        }
    }
});
```

在 html 标记中创建数据网格（DataGrid）

```
<table id="tt" style="width:600px;height:250px"
        url="data/datagrid_data.json" title="Editable DataGrid"
        singleSelect="true" idField="itemid" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="60">Item ID</th>
            <th field="listprice" width="80" align="right" edit
            <th field="unitcost" width="80" align="right" edito
            <th field="attr1" width="180" editor="text">Attribu
            <th field="status" width="60" align="center" editor
            <th field="action" width="80" align="center" format
        </tr>
    </thead>
</table>
```

我们分配 numberspinner 编辑器到 'unit cost' 字段。 当开始编辑一行，用户可以通过 numberspinner 编辑器来编辑数据。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid23.zip

jQuery EasyUI 数据网格 - 列运算

在本教程中，您将学习如何在可编辑的数据网格（datagrid）中包含一个运算的列。一个运算列通常包含一些从一个或多个其他列运算的值。

Editable DataGrid with Calculated Column

Item ID	List Price	Amount	Unit Cost	Attribute	Status
EST-1	16.5		10	Large	P
EST-10	18.5		12	Spotted Adult Female	P
EST-11	18.5		12	Venomless	P
EST-12	18.5		12	Rattleless	P
EST-13	18.5	2	37	Green Adult	P
EST-14	58.5	2	117	Tailless	☑
EST-15	23.5		12	With tail	P
EST-16	93.5		12	Adult Female	P
EST-17	93.5		12	Adult Male	P
EST-18	193.5		92	Adult Male	P

首先，创建一个可编辑的数据网格（datagrid）。这里我们创建了一些可编辑列，'listprice'、'amount' 和 'unitcost' 列定义为 numberbox 编辑类型。运算列是 'unitcost' 字段，将是 listprice 乘以 amount 列的结果。

```
    <table id="tt" style="width:600px;height:auto"
            title="Editable DataGrid with Calculated Column" iconCl
            idField="itemid" url="data/datagrid_data.json">
        <thead>
            <tr>
                <th field="itemid" width="80">Item ID</th>
                <th field="listprice" width="80" align="right" edit
                <th field="amount" width="80" align="right" editor=
                <th field="unitcost" width="80" align="right" edito
                <th field="attr1" width="150" editor="text">Attribu
                <th field="status" width="60" align="center" editor
            </tr>
        </thead>
    </table>
```

当用户点击一行的时候，我们开始一个编辑动作。

```
var lastIndex;
$('#tt').datagrid({
    onClickRow:function(rowIndex){
        if (lastIndex != rowIndex){
            $('#tt').datagrid('endEdit', lastIndex);
            $('#tt').datagrid('beginEdit', rowIndex);
            setEditing(rowIndex);
        }
        lastIndex = rowIndex;
    }
});
```

为了在一些列之间创建运算关系，我们应该得到当前的 editors，并绑定一些事件到它们上面。

```
function setEditing(rowIndex){
    var editors = $('#tt').datagrid('getEditors', rowIndex);
    var priceEditor = editors[0];
    var amountEditor = editors[1];
    var costEditor = editors[2];
    priceEditor.target.bind('change', function(){
        calculate();
    });
    amountEditor.target.bind('change', function(){
        calculate();
    });
    function calculate(){
        var cost = priceEditor.target.val() * amountEditor.targ
        $(costEditor.target).numberbox('setValue',cost);
    }
}
```

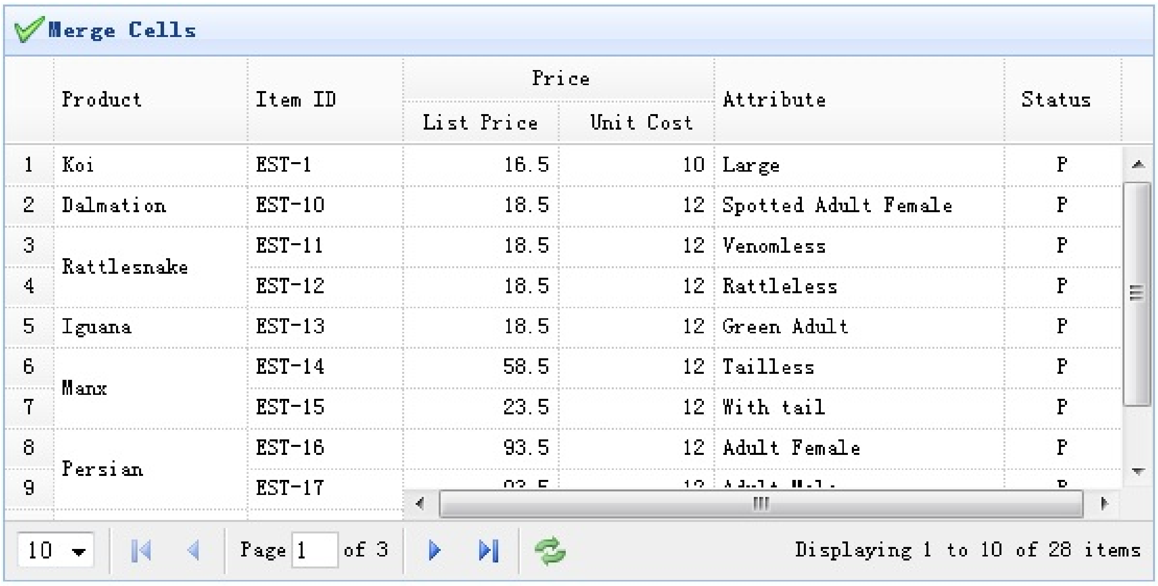
下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid15.zip

jQuery EasyUI 数据网格 - 合并单元格

数据网格（datagrid）经常需要合并一些单元格。本教程将向您展示如何在数据网格（datagrid）中合并单元格。

为了合并数据网格（datagrid）单元格，只需简单地调用 'mergeCells' 方法，并传入合并信息参数，告诉数据网格（datagrid）如何合并单元格。在所有合并的单元格中，除了第一个单元格，其它单元格在合并后被隐藏。



Merge Cells

	Product	Item ID	Price		Attribute	Status
			List Price	Unit Cost		
1	Koi	EST-1	16.5	10	Large	P
2	Dalmation	EST-10	18.5	12	Spotted Adult Female	P
3	Rattlesnake	EST-11	18.5	12	Venomless	P
4		EST-12	18.5	12	Rattleless	P
5	Iguana	EST-13	18.5	12	Green Adult	P
6	Manx	EST-14	58.5	12	Tailless	P
7		EST-15	23.5	12	With tail	P
8	Persian	EST-16	93.5	12	Adult Female	P
9		EST-17				

10 Page 1 of 3 Displaying 1 to 10 of 28 items

创建数据网格（DataGrid）

```
<table id="tt" title="Merge Cells" style="width:550px;height:25
        url="data/datagrid_data.json"
        singleSelect="true" iconCls="icon-save" rownumbers="tru
        idField="itemid" pagination="true">
    <thead frozen="true">
        <tr>
            <th field="productid" width="80" formatter="formatP
            <th field="itemid" width="100">Item ID</th>
        </tr>
    </thead>
    <thead>
        <tr>
            <th colspan="2">Price</th>
            <th rowspan="2" field="attr1" width="150">Attribute
            <th rowspan="2" field="status" width="60" align="ce
        </tr>
        <tr>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit 
        </tr>
    </thead>
</table>
```

合并单元格

当数据加载之后，我们合并数据网格（datagrid）中的一些单元格，所以放置下面的代码在 onLoadSuccess 回调函数中。

```
$('#tt').datagrid({
    onLoadSuccess:function(){
        var merges = [{
            index:2,
            rowspan:2
        },{
            index:5,
            rowspan:2
        },{
            index:7,
            rowspan:2
        }];
        for(var i=0; i<merges.length; i++)
            $('#tt').datagrid('mergeCells',{
                index:merges[i].index,
                field:'productid',
                rowspan:merges[i].rowspan
            });
    }
});
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid13.zip

jQuery EasyUI 数据网格 - 创建自定义视图

在不同的情况下，您可能需要为数据网格（datagrid）运用更灵活的布局。对于用户来说，卡片视图（Card View）是个不错的选择。这个工具可以在数据网格（datagrid）中迅速获取和显示数据。在数据网格（datagrid）的头部，您可以仅仅通过点击列的头部来排序数据。本教程将向您展示如何创建自定义卡片视图（Card View）。



创建卡片视图（Card View）

从数据网格（datagrid）的默认视图继承，是个创建自定义视图的不错方法。我们将要创建一个卡片视图（Card View）来为每行显示一些信息。

```
var cardview = $.extend({}, $.fn.datagrid.defaults.view, {
    renderRow: function(target, fields, frozen, rowIndex, rowDa
        var cc = [];
        cc.push('<td colspan=' + fields.length + ' style="paddi
        if (!frozen){
            var aa = rowData.itemid.split('-');
            var img = 'shirt' + aa[1] + '.gif';
            cc.push('<img src="images/' + img + '" style="widtl
            cc.push('<div style="float:left;margin-left:20px;">
            for(var i=0; i<fields.length; i++){
                var copts = $(target).datagrid('getColumnOptior
                cc.push('<p><span class="c-label">' + copts.tit
            }
            cc.push('</div>');
        }
        cc.push('</td>');
        return cc.join('');
    }
});
```

创建数据网格（DataGrid）

现在我们使用视图创建数据网格（datagrid）。

```
<table id="tt" style="width:500px;height:400px"
        title="DataGrid - CardView" singleSelect="true" fitColu
        url="datagrid8_getdata.php" pagination="true" sortOrder
    <thead>
        <tr>
            <th field="itemid" width="80" sortable="true">Item
            <th field="listprice" width="80" sortable="true">Li
            <th field="unitcost" width="80" sortable="true">Uni
            <th field="attr1" width="150" sortable="true">Attri
            <th field="status" width="60" sortable="true">Statu
        </tr>
    </thead>
</table>
```

```
$('#tt').datagrid({
    view: cardview
});
```

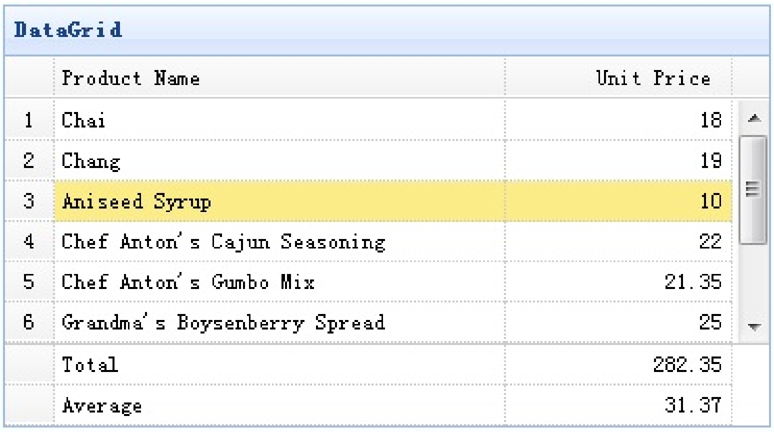
请注意，我们设置 view 属性，且它的值为我们的卡片视图。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid16.zip

jQuery EasyUI 数据网格 - 创建页脚摘要

在本教程中，我们将向您展示如何在数据网格（datagrid）页脚显示摘要信息行。



	Product Name	Unit Price
1	Chai	18
2	Chang	19
3	Aniseed Syrup	10
4	Chef Anton's Cajun Seasoning	22
5	Chef Anton's Gumbo Mix	21.35
6	Grandma's Boysenberry Spread	25
	Total	282.35
	Average	31.37

为了显示页脚行，您应该设置 showFooter 属性为 true，然后准备定义在数据网格（datagrid）数据中的页脚行。以下是示例数据：

```
{"total":1,"rows":[{"id":1,"name":"Chai","price":18.00}],"foote
```

创建数据网格（DataGrid）

```
<table id="tt" title="DataGrid" class="easyui-datagrid" style='
        url="data/datagrid17_data.json"
        fitColumns="true" rownumbers="true" showFooter="true">
    <thead>
        <tr>
            <th field="name" width="80">Product Name</th>
            <th field="price" width="40" align="right">Unit Pri
        </tr>
    </thead>
</table>
```

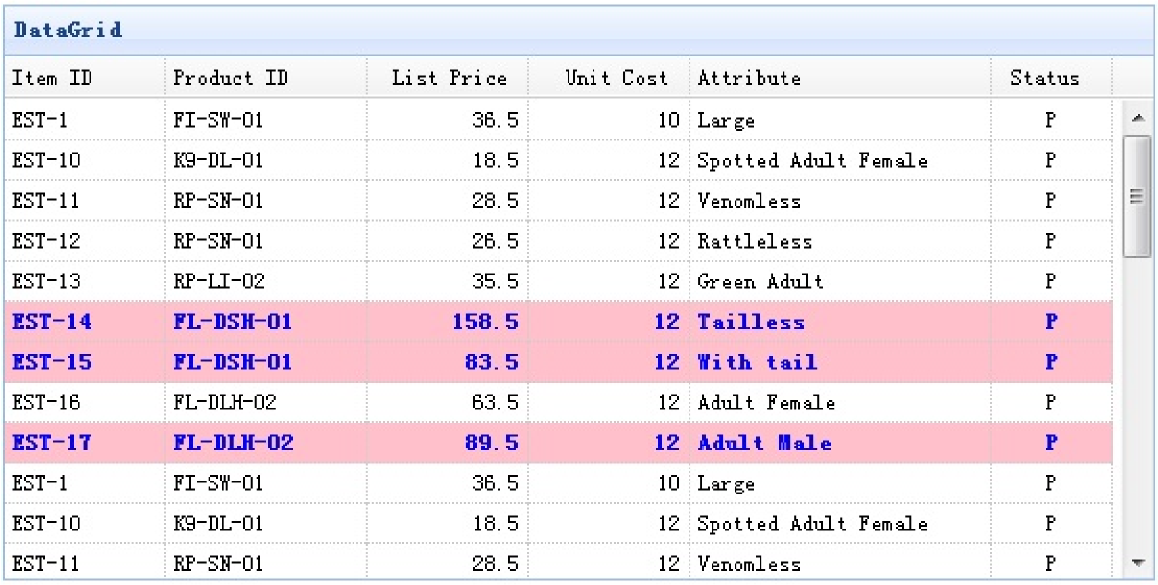
页脚行和显示数据行一样，所以您可以在页脚显示不止一个摘要信息。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid17.zip

jQuery EasyUI 数据网格 - 条件设置行背景颜色

本教程将向您展示如何根据一些条件改变数据网格（datagrid）组件的行样式。当 listprice 值大于 50 时，我们将为该行设置不同的颜色。



DataGrid

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	36.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	28.5	12	Venomless	P
EST-12	RP-SN-01	26.5	12	Rattleless	P
EST-13	RP-LI-02	35.5	12	Green Adult	P
EST-14	**FL-DSH-01**	**158.5**	**12**	**Tailless**	**P**
EST-15	**FL-DSH-01**	**83.5**	**12**	**With tail**	**P**
EST-16	FL-DLH-02	63.5	12	Adult Female	P
EST-17	**FL-DLH-02**	**89.5**	**12**	**Adult Male**	**P**
EST-1	FI-SW-01	36.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P
EST-11	RP-SN-01	28.5	12	Venomless	P

数据网格（datagrid）的 rowStyler 函数的设计目的是允许您自定义行样式。以下代码展示如何改变行样式：

```
<table id="tt" title="DataGrid" style="width:600px;height:250p
        url="data/datagrid_data.json"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" width="80" align="right">List
            <th field="unitcost" width="80" align="right">Unit
            <th field="attr1" width="150">Attribute</th>
            <th field="status" width="60" align="center">Stauts
        </tr>
    </thead>
</table>
```

```
$('#tt').datagrid({
    rowStyler:function(index,row){
        if (row.listprice>50){
            return 'background-color:pink;color:blue;font-weigh
        }
    }
});
```

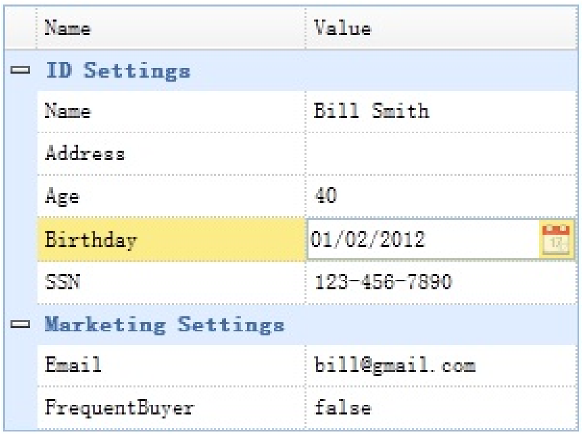
正如您所看到的，我们根据一些条件设置 background-color（背景色）为 pink（粉红色），设置文本颜色为 blue（蓝色）。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid18.zip

jQuery EasyUI 数据网格 - 创建属性网格

属性网格（property grid）带有一个内置的 expand（展开）/collapse（合并） 按钮，可以简单地为行分组。您可以简单地创建一个可编辑属性的分层（hierarchical）列表。



设置 HTML

```
<table id="tt" class="easyui-propertygrid" style="width:300px"
        url="propertygrid_data.json"
        showGroup="true" scrollbarSize="0"
></table>
```

准备 json 数据

```
[
    {"name":"Name","value":"Bill Smith","group":"ID Settings","
    {"name":"Address","value":"","group":"ID Settings","editor"
    {"name":"Age","value":"40","group":"ID Settings","editor":"
    {"name":"Birthday","value":"01/02/2012","group":"ID Setting
    {"name":"SSN","value":"123-456-7890","group":"ID Settings",
    {"name":"Email","value":"bill@gmail.com","group":"Marketing
        "type":"validatebox",
        "options":{
            "validType":"email"
        }
    }},
    {"name":"FrequentBuyer","value":"false","group":"Marketing
        "type":"checkbox",
        "options":{
            "on":true,
            "off":false
        }
    }}
]
```

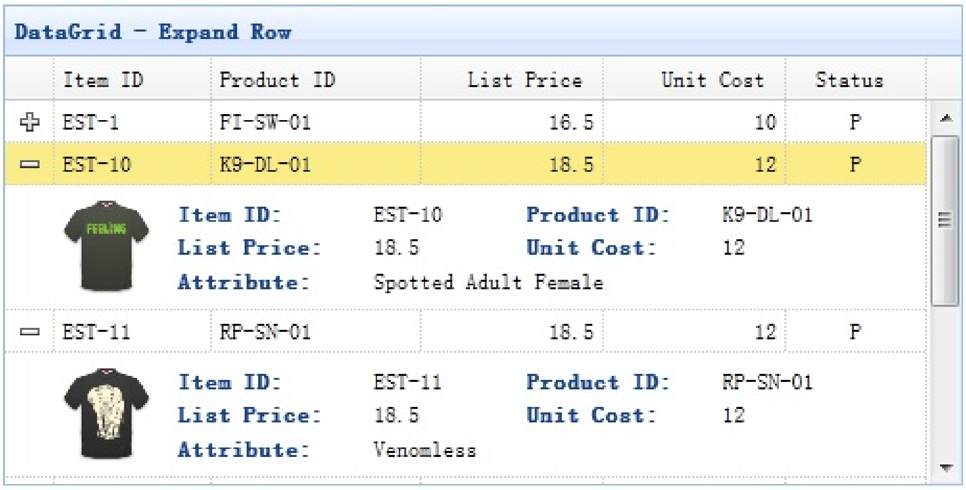
正如您所看到的，属性网格（property grid）的创建不需要任何的 javascript 代码。您可以简单地继承扩展 editor 类型。

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid20.zip

jQuery EasyUI 数据网格 - 扩展行显示细节

数据网格（datagrid）可以改变它的视图（view）来显示不同的效果。使用详细视图，数据网格（datagrid）可以在数据行的左边显示展开按钮（"+" 或者 "-"）。用户可以展开行来显示附加的详细信息。



步骤 1：创建数据网格（DataGrid）

```
<table id="dg" style="width:500px;height:250px"
        url="datagrid8_getdata.php"
        pagination="true" sortName="itemid" sortOrder="desc"
        title="DataGrid - Expand Row"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="60">Item ID</th>
            <th field="productid" width="80">Product ID</th>
            <th field="listprice" align="right" width="70">List
            <th field="unitcost" align="right" width="70">Unit
            <th field="status" width="50" align="center">Status
        </tr>
    </thead>
</table>
```

步骤 2：为数据网格（DataGrid）设置详细视图

为了使用详细视图，请记得在页面头部引用视图脚本文件。

```
<script type="text/javascript" src="http://www.w3cschool.cc/try/jea
```

```
$('#dg').datagrid({
    view: detailview,
    detailFormatter:function(index,row){
        return '<div class="ddv" style="padding:5px 0"></div>';
    },
    onExpandRow: function(index,row){
        var ddv = $(this).datagrid('getRowDetail',index).find('div
        ddv.panel({
            border:false,
            cache:false,
            href:'datagrid21_getdetail.php?itemid='+row.itemid,
            onLoad:function(){
                $('#dg').datagrid('fixDetailRowHeight',index);
            }
        });
        $('#dg').datagrid('fixDetailRowHeight',index);
    }
});
```

我们定义 'detailFormatter' 函数，告诉数据网格（datagrid）如何渲染详细视图。在这种情况下,我们返回一个简单的 '<div>' 元素，它将充当详细内容的容器。 请注意，详细信息为空。当用户点击展开按钮（'+'）时，onExpandRow 事件将被触发。 所以我们可以写一些代码来加载 ajax 详细内容。 最后我们调用 'fixDetailRowHeight' 方法来固定当详细内容加载时的行高度。

步骤 3：服务器端代码

datagrid21_getdetail.php

```
&lt;?php
    include_once 'conn.php';

    $itemid = mysql_real_escape_string($_REQUEST['itemid']);

    $rs = mysql_query("select * from item where itemid='$itemid'");
    $item = mysql_fetch_array($rs);
?&gt;

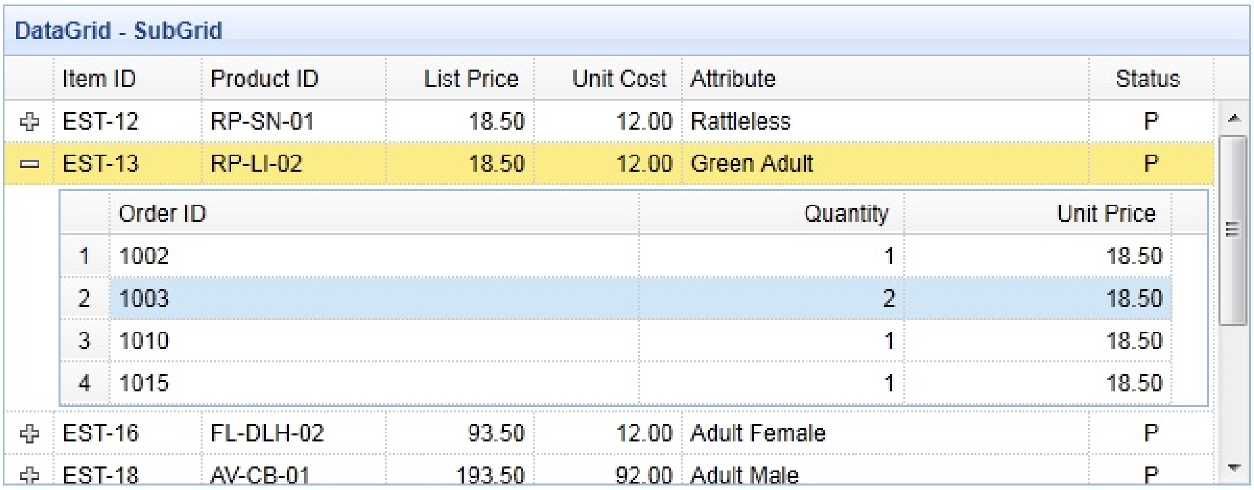
<table class="dv-table" border="0" style="width:100%;">
    <tr>
        <td rowspan="3" style="width:60px">
            &lt;?php
                $aa = explode('-',$itemid);
                $serno = $aa[1];
                $img = "images/shirt$serno.gif";
                echo "<img src=\"$img\" style=\"width:60px;margin-r
            ?&gt;
        </td>
        <td class="dv-label">Item ID: </td>
        <td>&lt;?php echo $item['itemid'];?&gt;</td>
        <td class="dv-label">Product ID:</td>
        <td>&lt;?php echo $item['productid'];?&gt;</td>
    </tr>
    <tr>
        <td class="dv-label">List Price: </td>
        <td>&lt;?php echo $item['listprice'];?&gt;</td>
        <td class="dv-label">Unit Cost:</td>
        <td>&lt;?php echo $item['unitcost'];?&gt;</td>
    </tr>
    <tr>
        <td class="dv-label">Attribute: </td>
        <td colspan="3">&lt;?php echo $item['attr1'];?&gt;</td>
    </tr>
</table>
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid21.zip

jQuery EasyUI 数据网格 - 创建子网格

使用数据网格（datagrid）的详细视图，用户可以展开一行来显示附加的详细信息。 任何内容都可以加载作为行详细，子网格也可以动态加载。 本教程将向您展示如何在主网格上创建一个子网格。



步骤 1：创建主网格

```
<table id="dg" style="width:700px;height:250px"
        url="datagrid22_getdata.php"
        title="DataGrid - SubGrid"
        singleSelect="true" fitColumns="true">
    <thead>
        <tr>
            <th field="itemid" width="80">Item ID</th>
            <th field="productid" width="100">Product ID</th>
            <th field="listprice" align="right" width="80">List Pri
            <th field="unitcost" align="right" width="80">Unit Cost
            <th field="attr1" width="220">Attribute</th>
            <th field="status" width="60" align="center">Status</th
        </tr>
    </thead>
</table>
```

步骤 2：设置详细视图来显示子网格

为了使用详细视图，请记得在页面头部引用视图脚本文件。

```
<script type="text/javascript" src="http://www.w3cschool.cc/try/jea
```

```
$('#dg').datagrid({
    view: detailview,
    detailFormatter:function(index,row){
        return '<div style="padding:2px"><table class="ddv"></table
    },
    onExpandRow: function(index,row){
        var ddv = $(this).datagrid('getRowDetail',index).find('tabl
        ddv.datagrid({
            url:'datagrid22_getdetail.php?itemid='+row.itemid,
            fitColumns:true,
            singleSelect:true,
            rownumbers:true,
            loadMsg:'',
            height:'auto',
            columns:[[
                {field:'orderid',title:'Order ID',width:100},
                {field:'quantity',title:'Quantity',width:100},
                {field:'unitprice',title:'Unit Price',width:100}
            ]],
            onResize:function(){
                $('#dg').datagrid('fixDetailRowHeight',index);
            },
            onLoadSuccess:function(){
                setTimeout(function(){
                    $('#dg').datagrid('fixDetailRowHeight',index);
                },0);
            }
        });
        $('#dg').datagrid('fixDetailRowHeight',index);
    }
});
```

当用户点击展开按钮（'+'）时，'onExpandRow' 事件将被触发。 我们创建一个新的带有三列的子网格。 当子网格数据加载成功时或者改变尺寸大小时，请记得对主网格调用 'fixDetailRowHeight' 方法。

步骤 3：服务器端代码

datagrid22_getdata.php

```
$result = array();

include 'conn.php';

$rs = mysql_query("select * from item where itemid in (select item

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}

echo json_encode($items);
```

datagrid22_getdetail.php

```
include 'conn.php';

$itemid = mysql_real_escape_string($_REQUEST['itemid']);

$rs = mysql_query("select * from lineitem where itemid='$itemid'");
$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
echo json_encode($items);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid22.zip

jQuery EasyUI 数据网格 - 使用虚拟滚动视图显示海量数据

数据网格（datagrid）的虚拟滚动特性可以用来显示大数量的记录而不需要分页。当滚动垂直滚动条时，数据网格（datagrid）执行 ajax 请求来加载和刷新现有的记录。 整个刷新的行为过程是平稳的没有闪烁。 在本教程中，我们将创建一个数据网格（datagrid），并运用虚拟滚动特性从服务器加载数据。

DataGrid - VirtualScrollView

	Inv No	Date	Name	Amount	Price	Cost	Note
6279	INV6279	2012-10-05	Name29	94	113.01	10622.94	Note29
6280	INV6280	2012-10-06	Name30	97	196.41	19051.77	Note30
6281	INV6281	2012-10-07	Name31	81	111.97	9069.57	Note31
6282	INV6282	2012-10-08	Name32	81	181.21	14678.01	Note32
6283	INV6283	2012-10-09	Name33	68	159.96	10877.28	Note33
6284	INV6284	2012-10-10	Name34	87	113.47	9871.89	Note34
6285	INV6285	2012-10-11	Name35	92	191.35	17604.20	Note35
6286	INV6286	2012-10-12	Name36	88	102.56	9025.28	Note36
6287	INV6287	2012-10-13	Name37	63	190.51	12002.13	Note37
6288	INV6288	2012-10-14	Name38	83	107.88	8954.04	Note38

创建数据网格（DataGrid）

为数据网格（datagrid）运用虚拟滚动特性，'view' 属性应该设置为 'scrollview'。用户应该从数据网格（datagrid）扩展下载 scrollview，并在页面头部引用 scrollview 文件。

```
<script type="text/javascript" src="http://www.w3cschool.cc/try/jea
```

```
<table id="tt" class="easyui-datagrid" style="width:700px;height:3(
        title="DataGrid - VirtualScrollView"
        data-options="view:scrollview,rownumbers:true,singleSelect
            url:'datagrid27_getdata.php',autoRowHeight:false,pageS:
    <thead>
        <tr>
            <th field="inv" width="80">Inv No</th>
            <th field="date" width="100">Date</th>
            <th field="name" width="80">Name</th>
            <th field="amount" width="80" align="right">Amount</th>
            <th field="price" width="80" align="right">Price</th>
            <th field="cost" width="100" align="right">Cost</th>
            <th field="note" width="110">Note</th>
        </tr>
    </thead>
</table>
```

请注意，这里我们不需要使用 pagination 属性，但 pageSize 属性是必需的，这样执行 ajax 请求时，数据网格（datagrid）将从服务器获取指定数量的记录。

服务器端代码

datagrid27_getdata.php

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 50;

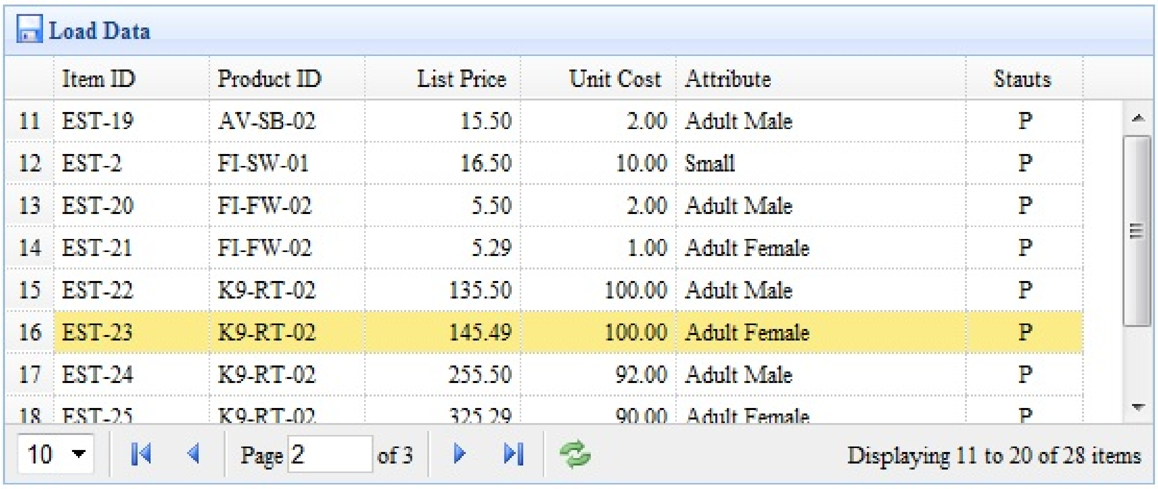
$items = array();
date_default_timezone_set('UTC');
for($i=1; $i<=$rows; $i++){
    $index = $i+($page-1)*$rows;
    $amount = rand(50,100);
    $price = rand(10000,20000)/100;
    $items[] = array(
        'inv' => sprintf("INV%04d",$index),
        'date' => date('Y-m-d',time()+24*3600*$i),
        'name' => 'Name' . $index,
        'note' => 'Note' . $index,
        'amount' => $amount,
        'price' => sprintf('%01.2f',$price),
        'cost' => sprintf('%01.2f',$amount*$price)
    );
}
$result = array();
$result['total'] = 8000;
$result['rows'] = $items;
echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid27.zip

jQuery EasyUI 数据网格 - 添加分页组件

本实例演示如何从服务器端加载数据，如何添加分页组件（pagination）到数据网格（datagrid）。



创建数据网格（DataGrid）

为了从远程服务器端加载数据，您应该设置 'url' 属性，在您的服务器端应该返回 JSON 格式数据。请看数据网格（datagrid）文档得到更多关于它的数据格式信息。

```
    <table id="tt" class="easyui-datagrid" style="width:600px;heigh
            url="datagrid2_getdata.php"
            title="Load Data" iconCls="icon-save"
            rownumbers="true" pagination="true">
        <thead>
            <tr>
                <th field="itemid" width="80">Item ID</th>
                <th field="productid" width="80">Product ID</th>
                <th field="listprice" width="80" align="right">List
                <th field="unitcost" width="80" align="right">Unit
                <th field="attr1" width="150">Attribute</th>
                <th field="status" width="60" align="center">Stauts
            </tr>
        </thead>
    </table>
```

我们定义数据网格（datagrid）列，并设置 'pagination' 属性为 true，它将在数据网格（datagrid）的底部生成一个分页（pagination）工具栏。pagination将发送两个参数到服务器：

- page：页码，起始值 1。

- rows：每页显示行。

服务器端代码

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
// ...
$rs = mysql_query("select count(*) from item");
$row = mysql_fetch_row($rs);
$result["total"] = $row[0];

$rs = mysql_query("select * from item limit $offset,$rows");

$items = array();
while($row = mysql_fetch_object($rs)){
    array_push($items, $row);
}
$result["rows"] = $items;

echo json_encode($result);
```

下载 jQuery EasyUI 实例

jeasyui-datagrid-datagrid2.zip

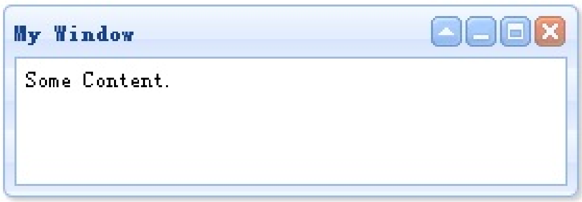
jQuery EasyUI 窗口

jQuery EasyUI 窗口 - 创建简单窗口

创建一个窗口（window）非常简单，我们创建一个 DIV 标记：

```
<div id="win" class="easyui-window" title="My Window" style="width
    Some Content.
</div>
```

现在运行测试页面，您会看见一个窗口（window）显示在您的屏幕上。我们不需要写任何的 javascript 代码。



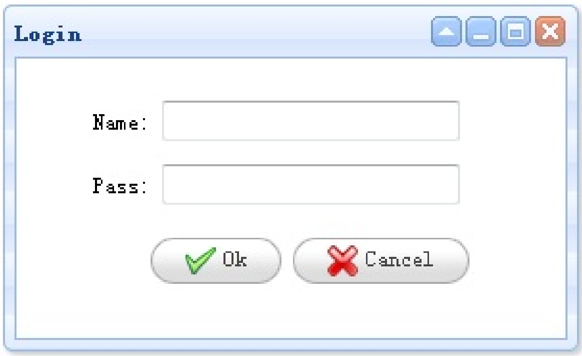
如果您希望创建一个隐藏的窗口（window），记得设置 'closed' 属性为 'true' 值，您可以调用 'open' 方法来打开窗口（window）：

```
<div id="win" class="easyui-window" title="My Window" closed="true'
    Some Content.
</div>
```

```
$('#win').window('open');
```

作为最后的实例演示，我们创建一个登录窗口（window）：

```
<div id="win" class="easyui-window" title="Login" style="width:300p
    <form style="padding:10px 20px 10px 40px;">
        <p>Name: <input type="text"></p>
        <p>Pass: <input type="password"></p>
        <div style="padding:5px;text-align:center;">
            <a href="#" class="easyui-linkbutton" icon="icon-ok">Ok
            <a href="#" class="easyui-linkbutton" icon="icon-cancel
        </div>
    </form>
</div>
```



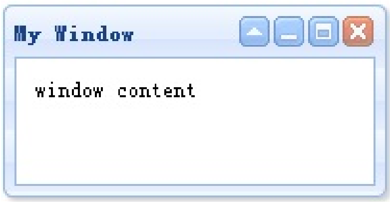
下载 jQuery EasyUI 实例

jeasyui-win-win1.zip

jQuery EasyUI 窗口 - 自定义窗口工具栏

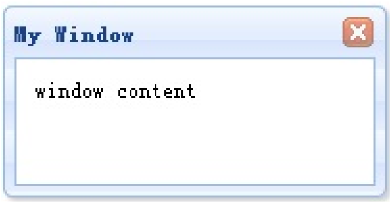
默认情况下，窗口（window）有四个工具：collapsible、minimizable、maximizable 和 closable。比如我们定义以下窗口（window）：

```
<div id="win" class="easyui-window" title="My Window" style="pa
    window content
</div>
```



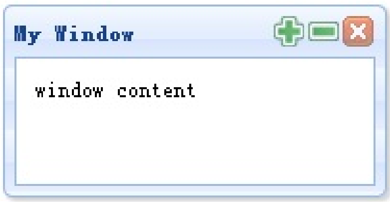
如需自定义工具，设置该工具为 true 或者 false。比如我们希望定义一个窗口（window），仅仅拥有一个可关闭的工具。您应该设置任何其他工具为 false。我们可以在标记中或者通过 jQuery 代码定义 tools 属性。现在我们使用 jQuery 代码来定义窗口（window）：

```
$('#win').window({
    collapsible:false,
    minimizable:false,
    maximizable:false
});
```



如果我们希望添加自定义的工具到窗口（window），我们可以使用 tools 属性。作为实例演示，我们添加两个工具到窗口（window）：

```
$('#win').window({
    collapsible:false,
    minimizable:false,
    maximizable:false,
    tools:[{
        iconCls:'icon-add',
        handler:function(){
            alert('add');
        }
    },{
        iconCls:'icon-remove',
        handler:function(){
            alert('remove');
        }
    }]
});
```



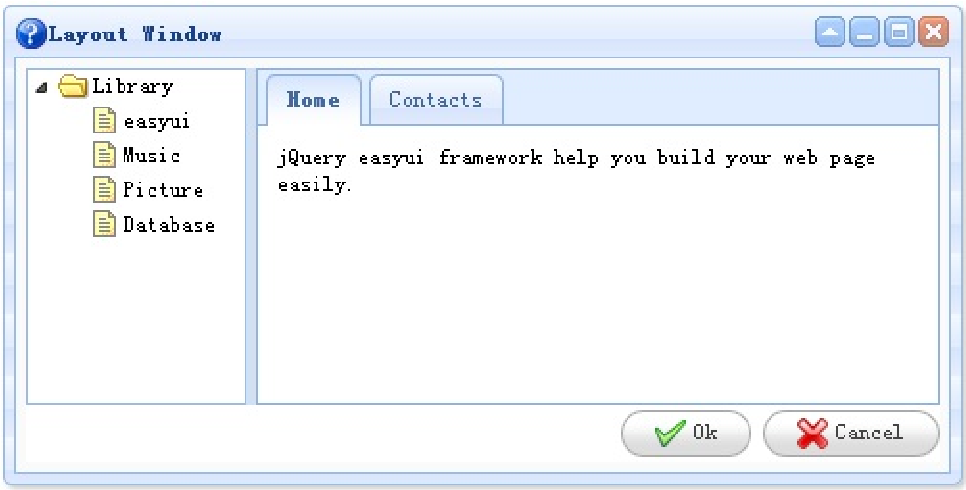
下载 jQuery EasyUI 实例

jeasyui-win-win2.zip

jQuery EasyUI 窗口 - 窗口与布局

Layout 组件可以内嵌在窗口（window）中。我们可以创建一个复杂的布局窗口，甚至不需要写任何的 js 代码。jquery-easyui 框架帮我们在后台做渲染和调整尺寸。

作为一个实例，我们创建一个窗口（window），它包含两个部分，一个放置在左边一个放置在右边。在窗口（window）的左边我们创建一个树形菜单（tree），在窗口（window）的右边我们创建一个 tabs 容器。



```
    <div class="easyui-window" title="Layout Window" icon="icon-hel
        <div class="easyui-layout" fit="true">
            <div region="west" split="true" style="width:120px;">
                <ul class="easyui-tree">
                    <li>
                        <span>Library</span>
                        <ul>
                            <li><span>easyui</span></li>
                            <li><span>Music</span></li>
                            <li><span>Picture</span></li>
                            <li><span>Database</span></li>
                        </ul>
                    </li>
                </ul>
            </div>
            <div region="center" border="false" border="false">
                <div class="easyui-tabs" fit="true">
                    <div title="Home" style="padding:10px;">
                        jQuery easyui framework help you build your
                    </div>
                    <div title="Contacts">
                        No contact data.
                    </div>
                </div>
            </div>
            <div region="south" border="false" style="text-align:ri
                <a class="easyui-linkbutton" icon="icon-ok" href="j
                <a class="easyui-linkbutton" icon="icon-cancel" hre
            </div>
        </div>
    </div>
```

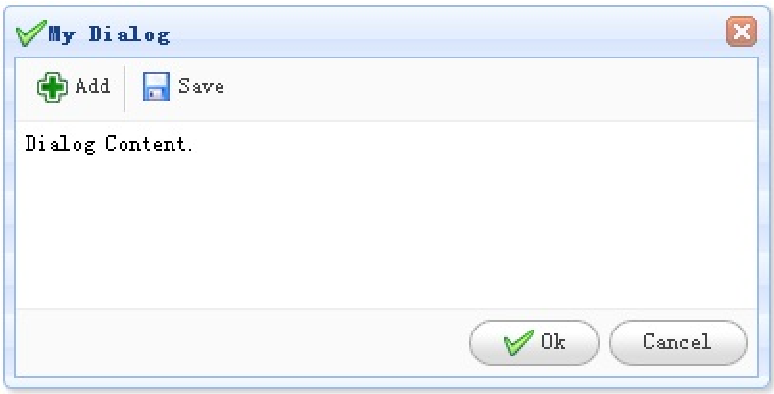
请看上面的代码，我们仅仅使用了 HTML 标记，一个复杂的布局窗口（window）将显示。这就是 jquery-easyui 框架，简单而强大。

下载 jQuery EasyUI 实例

jeasyui-win-win3.zip

jQuery EasyUI 窗口 - 创建对话框

对话框（Dialog）是一个特殊的窗口（window），可以包含在顶部的工具栏和在底部的按钮。默认情况下，对话框（Dialog）不能改变大小，但是用户可以设置 resizable 属性为 true，使其可以改变大小。



创建对话框（Dialog）

对话框（Dialog）非常简单，可以从 DIV 标记创建，如下所示：

```
<div id="dd" class="easyui-dialog" style="padding:5px;width:40(
        title="My Dialog" iconCls="icon-ok"
        toolbar="#dlg-toolbar" buttons="#dlg-buttons">
    Dialog Content.
</div>
```

准备工具栏（Toolbar）和按钮（Button）

```
<div id="dlg-toolbar">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" pl
    <a href="#" class="easyui-linkbutton" iconCls="icon-save" p
</div>
<div id="dlg-buttons">
    <a href="#" class="easyui-linkbutton" iconCls="icon-ok" onc
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel'
</div>
```

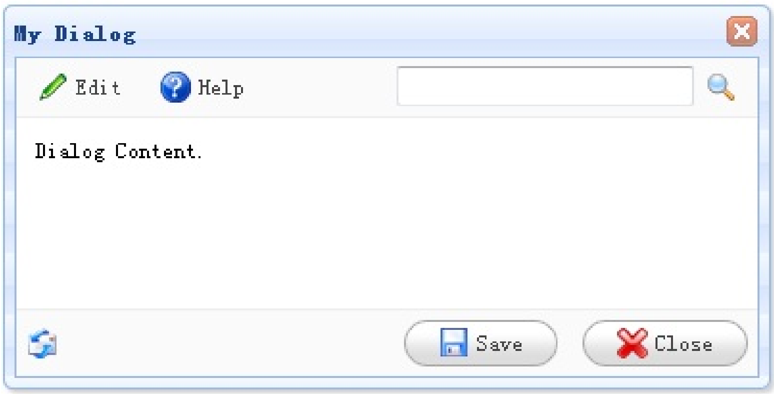
上面的代码我们创建了一个带有工具栏（toolbar）和按钮（button）的对话框（dialog）。这是对话框（dialog）、工具栏（toolbar）、内容（content）和按钮（buttons）的标准配置。

下载 jQuery EasyUI 实例

jeasyui-win-dlg1.zip

jQuery EasyUI 窗口 - 自定义带有工具条和按钮的对话框

您可以创建一个带有工具栏（toolbar）和按钮（button）的对话框（dialog），可以从 HTML 标记创建。这个教程描述如何添加工具栏（toolbar）和按钮（button）到对话框（dialog），没有任何的 javascript 代码。



创建对话框（Dialog）

```
<div id="dd" class="easyui-dialog" title="My Dialog" style="wic
        toolbar="#dlg-toolbar" buttons="#dlg-buttons">
    Dialog Content.
</div>
```

创建工具栏（Toolbar）

```
<div id="dlg-toolbar">
    <table cellpadding="0" cellspacing="0" style="width:100%">
        <tr>
            <td>
                <a href="#" class="easyui-linkbutton" iconCls='
                <a href="#" class="easyui-linkbutton" iconCls='
            </td>
            <td style="text-align:right">
                <input></input><a href="#" class="easyui-linkbu
            </td>
        </tr>
    </table>
</div>
```

创建按钮（Buttons）

```
<div id="dlg-buttons">
    <table cellpadding="0" cellspacing="0" style="width:100%">
        <tr>
            <td>
                <img src="email.gif"/>
            </td>
            <td style="text-align:right">
                <a href="#" class="easyui-linkbutton" iconCls='
                <a href="#" class="easyui-linkbutton" iconCls='
            </td>
        </tr>
    </table>
</div>
```

请注意，对话框（dialog）的工具栏（toolbar）和按钮（buttons）属性也可以通过 string 值指定，它将充当作为一个选择器去选择一个适当的 DIV 元素，并追加到工具栏（toolbar）或者按钮（buttons）的位置。

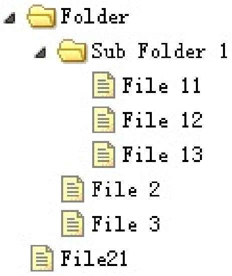
下载 jQuery EasyUI 实例

jeasyui-win-dlg2.zip

jQuery EasyUI 树形菜单

jQuery EasyUI 树形菜单 - 使用标记创建树形菜单

一个树形菜单（Tree）可以从标记创建。easyui 树形菜单（Tree）也可以定义在 <ul> 元素中。无序列表的 <ul> 元素提供一个基础的树（Tree）结构。每一个 <li> 元素将产生一个树节点，子 <ul> 元素将产生一个父树节点。



创建树形菜单（Tree）

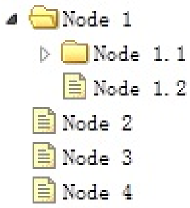
```
<ul class="easyui-tree">
    <li>
        <span>Folder</span>
        <ul>
            <li>
                <span>Sub Folder 1</span>
                <ul>
                    <li><span>File 11</span></li>
                    <li><span>File 12</span></li>
                    <li><span>File 13</span></li>
                </ul>
            </li>
            <li><span>File 2</span></li>
            <li><span>File 3</span></li>
        </ul>
    </li>
    <li><span>File21</span></li>
</ul>
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree1.zip

jQuery EasyUI 树形菜单 - 创建异步树形菜单

为了创建异步的树形菜单（Tree），每一个树节点必须要有一个 'id' 属性，这个将提交回服务器去检索子节点数据。



创建树形菜单（Tree）

```
<ul id="tt" class="easyui-tree"
        url="tree2_getdata.php">
</ul>
```

服务器端代码

```
$id = isset($_POST['id']) ? intval($_POST['id']) : 0;

include 'conn.php';

$result = array();
$rs = mysql_query("select * from nodes where parentId=$id");
while($row = mysql_fetch_array($rs)){
    $node = array();
    $node['id'] = $row['id'];
    $node['text'] = $row['name'];
    $node['state'] = has_child($row['id']) ? 'closed' : 'open'
    array_push($result,$node);
}

echo json_encode($result);

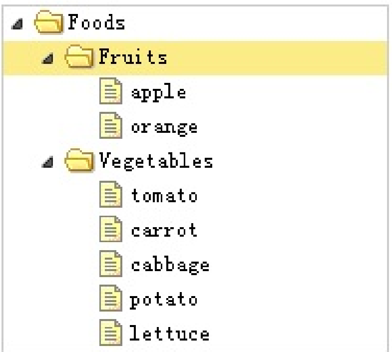
function has_child($id){
    $rs = mysql_query("select count(*) from nodes where parentI
    $row = mysql_fetch_array($rs);
    return $row[0] > 0 ? true : false;
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree2.zip

jQuery EasyUI 树形菜单 - 树形菜单添加节点

本教程向您展示如何附加节点到树形菜单（Tree）。我们将创建一个包含水果和蔬菜节点的食品树，然后添加一些其他水果到已存在的水果节点。



创建食品树

首先，我们创建食品树，代码如下所示：

```
<div style="width:200px;height:auto;border:1px solid #ccc;">
    <ul id="tt" class="easyui-tree" url="tree_data.json"></ul>
</div>
```

请注意，树（Tree)组件是定义在 <ul> 标记中，树节点数据从 URL "tree_data.json" 加载。

得到父节点

然后我们通过点击节点选择水果节点，我们将添加一些其他的水果数据。执行 getSelected 方法得到处理节点：

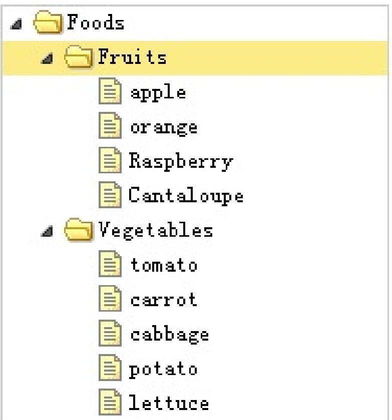
```
var node = $('#tt').tree('getSelected');
```

getSelected 方法的返回结果是一个 javascript 对象，它有一个 id、text、target 属性。target 属性是一个 DOM 对象，引用选中节点，它的 append 方法将用于附加子节点。

附加节点

```
var node = $('#tt').tree('getSelected');
if (node){
    var nodes = [{
        "id":13,
        "text":"Raspberry"
    },{
        "id":14,
        "text":"Cantaloupe"
    }];
    $('#tt').tree('append', {
        parent:node.target,
        data:nodes
    });
}
```

当添加一些水果，您将看见：



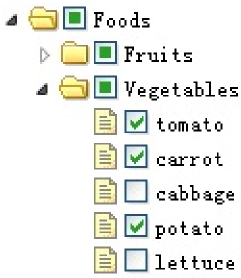
正如您所看到的，使用 easyui 的树（Tree）插件去附加节点不是那么的难。

下载 jQuery EasyUI 实例

jeasyui-tree-tree3.zip

jQuery EasyUI 树形菜单 - 创建带复选框的树形菜单

easyui 的树（Tree）插件允许您创建一个复选框树。如果您点击一个节点的复选框，这个点击的节点信息将向上和向下继承。例如：点击 'tomato' 节点的复选框，您将会看见 'Vegetables' 节点现在仅仅选中部分。



创建复选框树

```
<ul id="tt" class="easyui-tree"
        url="data/tree_data.json"
        checkbox="true">
</ul>
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree4.zip

jQuery EasyUI 树形菜单 - 树形菜单拖放控制

当在一个应用中使用树（Tree）插件，拖拽（drag）和放置（drop）功能要求允许用户改变节点位置。启用拖拽（drag）和放置（drop）操作，所有您需要做的就是把树（Tree）插件的 'dnd' 属性设置为 true。



创建树形菜单（Tree）

```
$('#tt').tree({
    dnd: true,
    url: 'tree_data.json'
});
```

当在一个树节点上发生放置操作，'onDrop' 事件将被触发，您应该做一些或更多的操作，例如保存节点状态到远程服务器端，等等。

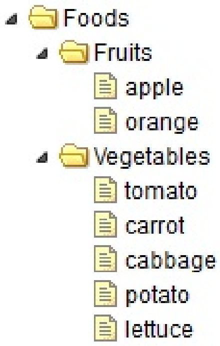
```
onDrop: function(targetNode, source, point){
    var targetId = $(target).tree('getNode', targetNode).id;
    $.ajax({
        url: '...',
        type: 'post',
        dataType: 'json',
        data: {
            id: source.id,
            targetId: targetId,
            point: point
        }
    });
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-tree5.zip

jQuery EasyUI 树形菜单 - 树形菜单加载父/子节点

通常表示一个树节点的方式就是在每一个节点存储一个 parentid。 这个也被称为邻接列表模型。 直接加载这些数据到树形菜单（Tree）是不允许的。 但是我们可以在加载树形菜单之前，把它转换为标准标准的树形菜单（Tree）数据格式。 树（Tree）插件提供一个 'loadFilter' 选项函数，它可以实现这个功能。 它提供一个机会来改变任何一个进入数据。 本教程向您展示如何使用 'loadFilter' 函数加载父/子节点到树形菜单（Tree）。



父/子节点数据

```
[
{"id":1,"parendId":0,"name":"Foods"},
{"id":2,"parentId":1,"name":"Fruits"},
{"id":3,"parentId":1,"name":"Vegetables"},
{"id":4,"parentId":2,"name":"apple"},
{"id":5,"parentId":2,"name":"orange"},
{"id":6,"parentId":3,"name":"tomato"},
{"id":7,"parentId":3,"name":"carrot"},
{"id":8,"parentId":3,"name":"cabbage"},
{"id":9,"parentId":3,"name":"potato"},
{"id":10,"parentId":3,"name":"lettuce"}
]
```

使用 'loadFilter' 创建树形菜单（Tree）

```
$('#tt').tree({
    url: 'data/tree6_data.json',
    loadFilter: function(rows){
        return convert(rows);
    }
});
```

转换的实现

```
    function convert(rows){
        function exists(rows, parentId){
            for(var i=0; i<rows.length; i++){
                if (rows[i].id == parentId) return true;
            }
            return false;
        }
        
        var nodes = [];
        // get the top level nodes
        for(var i=0; i<rows.length; i++){
            var row = rows[i];
            if (!exists(rows, row.parentId)){
                nodes.push({
                    id:row.id,
                    text:row.name
                });
            }
        }
        
        var toDo = [];
        for(var i=0; i<nodes.length; i++){
            toDo.push(nodes[i]);
        }
        while(toDo.length){
            var node = toDo.shift();    // the parent node
            // get the children nodes
            for(var i=0; i<rows.length; i++){
                var row = rows[i];
                if (row.parentId == node.id){
                    var child = {id:row.id,text:row.name};
                    if (node.children){
                        node.children.push(child);
                    } else {
                        node.children = [child];
                    }
                    toDo.push(child);
                }
            }
        }
        return nodes;
    }
```

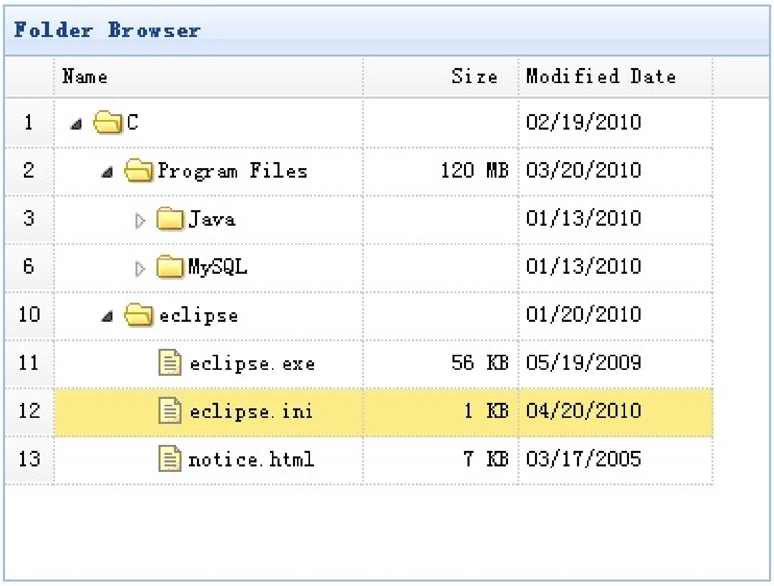
下载 jQuery EasyUI 实例

jeasyui-tree-tree6.zip

jQuery EasyUI 树形菜单 - 创建基础树形网格

树形网格（TreeGrid）组件从数据网格（DataGrid）继承，但是允许在行之间存在父/子节点关系。许多属性继承至数据网格（DataGrid），可以用在树形网格（TreeGrid）中。为了使用树形网格（TreeGrid），用户必须定义 'treeField' 属性，指明哪个字段作为树节点。

本教程将向您展示如何使用树形网格（TreeGrid）组件设置一个文件夹浏览。



创建树形网格（TreeGrid）

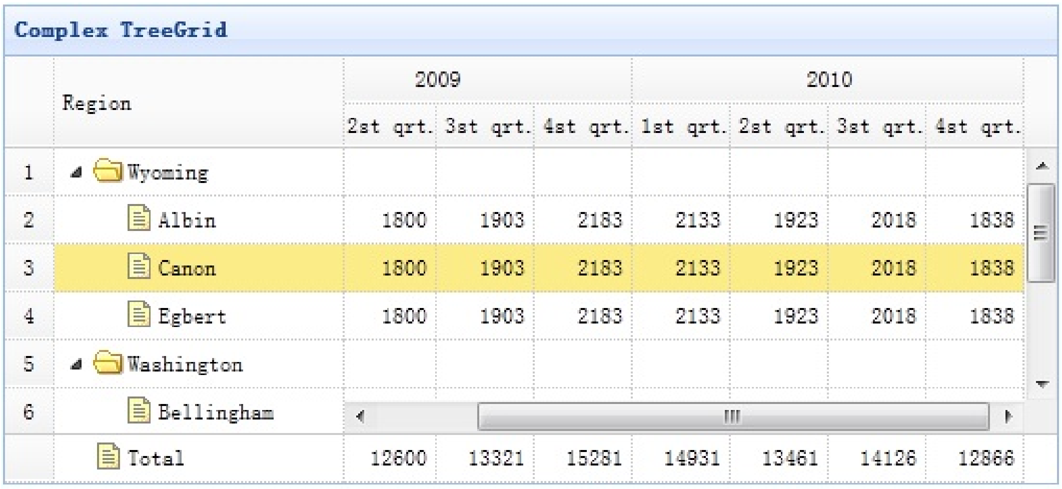
```
    <table id="test" title="Folder Browser" class="easyui-treegrid"
            url="data/treegrid_data.json"
            rownumbers="true"
            idField="id" treeField="name">
        <thead>
            <tr>
                <th field="name" width="160">Name</th>
                <th field="size" width="60" align="right">Size</th>
                <th field="date" width="100">Modified Date</th>
            </tr>
        </thead>
    </table>
```

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid1.zip

jQuery EasyUI 树形菜单 - 创建复杂树形网格

树形网格（TreeGrid）可以展示有限空间上带有多列和复杂数据电子表格。本教程将演示如何将表格数据排列在分割的网格和多行表头中，以便组织共同的数据。



Complex TreeGrid

	Region	2009			2010			
		2st qrt.	3st qrt.	4st qrt.	1st qrt.	2st qrt.	3st qrt.	4st qrt.
1	Wyoming							
2	Albin	1800	1903	2183	2133	1923	2018	1838
3	Canon	1800	1903	2183	2133	1923	2018	1838
4	Egbert	1800	1903	2183	2133	1923	2018	1838
5	Washington							
6	Bellingham							
	Total	12600	13321	15281	14931	13461	14126	12866

创建树形网格（TreeGrid）

```
    <table title="Complex TreeGrid" class="easyui-treegrid" style='
            url="data/treegrid2_data.json"
            rownumbers="true" showFooter="true"
            idField="id" treeField="region">
        <thead frozen="true">
            <tr>
                <th field="region" width="150">Region</th>
            </tr>
        </thead>
        <thead>
            <tr>
                <th colspan="4">2009</th>
                <th colspan="4">2010</th>
            </tr>
            <tr>
                <th field="f1" width="50" align="right">1st qrt.</t
                <th field="f2" width="50" align="right">2st qrt.</t
                <th field="f3" width="50" align="right">3st qrt.</t
                <th field="f4" width="50" align="right">4st qrt.</t
                <th field="f5" width="50" align="right">1st qrt.</t
                <th field="f6" width="50" align="right">2st qrt.</t
                <th field="f7" width="50" align="right">3st qrt.</t
                <th field="f8" width="50" align="right">4st qrt.</t
            </tr>
        </thead>
    </table>
```

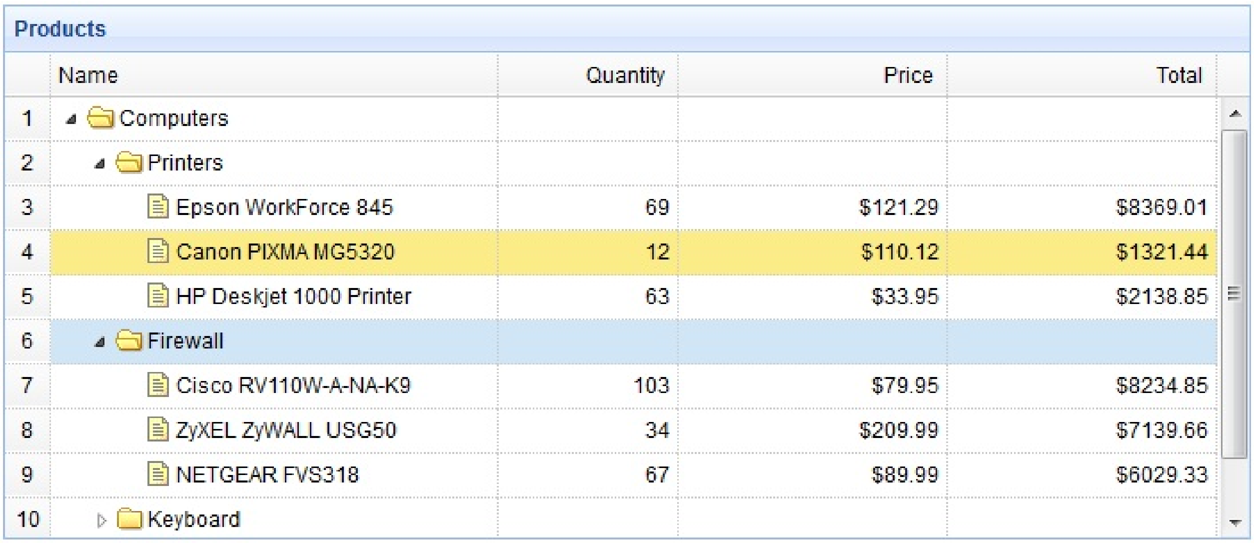
正如您所看到的，树形网格（Treegrid）的使用和数据网格（Datagrid）一样。请使用 'frozen' 属性来定义冻结列，列的 'colspan' 和 'rowspan' 属性来定义多行表头。

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid2.zip

jQuery EasyUI 树形菜单 - 树形网格动态加载

动态加载树形网格有助于从服务器上加载部分的行数据，避免加载大型数据的长时间等待。本教程将向您展示如何创建带有动态加载特性的树形网格（TreeGrid）。



创建树形网格（TreeGrid）

```
<table title="Products" class="easyui-treegrid" style="width:70
        url="treegrid3_getdata.php"
        rownumbers="true"
        idField="id" treeField="name">
    <thead>
        <tr>
            <th field="name" width="250">Name</th>
            <th field="quantity" width="100" align="right">Quan
            <th field="price" width="150" align="right" formatt
            <th field="total" width="150" align="right" formatt
        </tr>
    </thead>
</table>
```

服务器端代码

treegrid3_getdata.php

```
$id = isset($_POST['id']) ? intval($_POST['id']) : 0;

include 'conn.php';
$result = array();
$rs = mysql_query("select * from products where parentId=$id");
while($row = mysql_fetch_array($rs)){
    $row['state'] = has_child($row['id']) ? 'closed' : 'open';
    $row['total'] = $row['price']*$row['quantity'];
    array_push($result, $row);
}

echo json_encode($result);

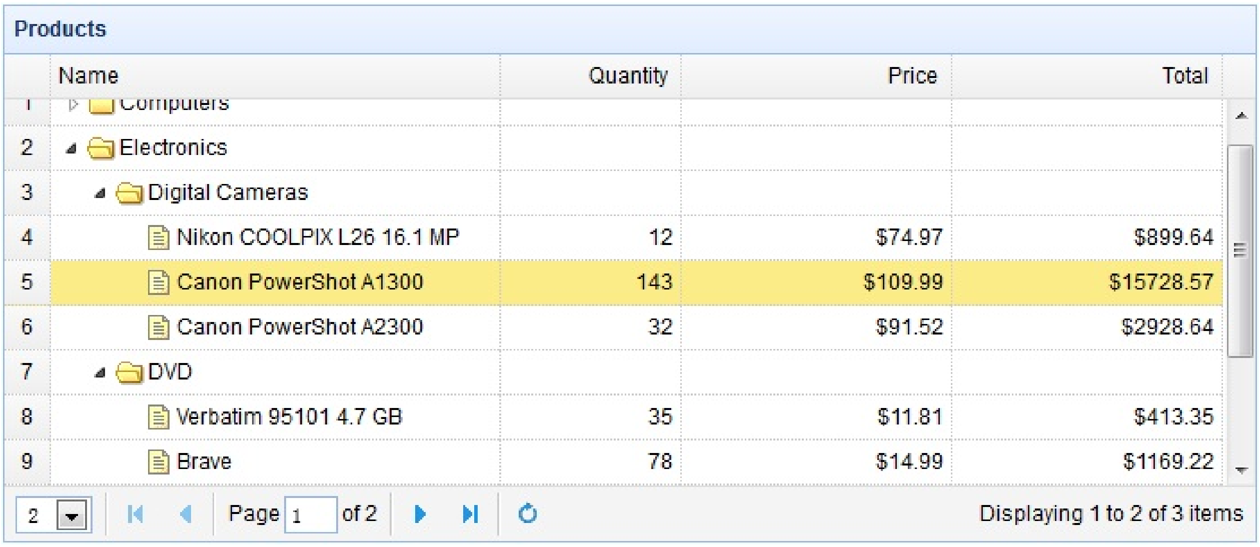
function has_child($id){
    $rs = mysql_query("select count(*) from products where parentI
    $row = mysql_fetch_array($rs);
    return $row[0] > 0 ? true : false;
}
```

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid3.zip

jQuery EasyUI 树形菜单 - 树形网格添加分页

本教程展示如何向带有动态加载特性的树形网格（TreeGrid）添加分页。



创建树形网格（TreeGrid）

启用树形网格（TreeGrid）的分页特性，必须添加 'pagination:true' 属性，这样页面加载时就会向服务器发送 'page' 和 'rows' 参数。

```
    <table title="Products" class="easyui-treegrid" style="width:7(
            data-options="
                url: 'treegrid4_getdata.php',
                rownumbers: true,
                pagination: true,
                pageSize: 2,
                pageList: [2,10,20],
                idField: 'id',
                treeField: 'name',
                onBeforeLoad: function(row,param){
                    if (!row) {    // load top level rows
                        param.id = 0;    // set id=0, indicate to l
                    }
                }
            ">
        <thead>
            <tr>
                <th field="name" width="250">Name</th>
                <th field="quantity" width="100" align="right">Quar
                <th field="price" width="150" align="right" formatt
                <th field="total" width="150" align="right" formatt
            </tr>
        </thead>
    </table>
```

服务器端代码

treegrid4_getdata.php

```
$page = isset($_POST['page']) ? intval($_POST['page']) : 1;
$rows = isset($_POST['rows']) ? intval($_POST['rows']) : 10;
$offset = ($page-1)*$rows;

$id = isset($_POST['id']) ? intval($_POST['id']) : 0;

include 'conn.php';

$result = array();
if ($id == 0){
    $rs = mysql_query("select count(*) from products where parentI
    $row = mysql_fetch_row($rs);
    $result["total"] = $row[0];

    $rs = mysql_query("select * from products where parentId=0 lim
    $items = array();
    while($row = mysql_fetch_array($rs)){
        $row['state'] = has_child($row['id']) ? 'closed' : 'open';
        array_push($items, $row);
    }
    $result["rows"] = $items;
} else {
    $rs = mysql_query("select * from products where parentId=$id")
    while($row = mysql_fetch_array($rs)){
        $row['state'] = has_child($row['id']) ? 'closed' : 'open';
        $row['total'] = $row['price']*$row['quantity'];
        array_push($result, $row);
    }
}

echo json_encode($result);

function has_child($id){
    $rs = mysql_query("select count(*) from products where parentI
    $row = mysql_fetch_array($rs);
    return $row[0] > 0 ? true : false;
}
```

发送到服务器的参数包括：

- page：要加载的当前页面。
- rows：页面尺寸大小。
- id：父行的 id 值，从服务器返回的行将被添加。

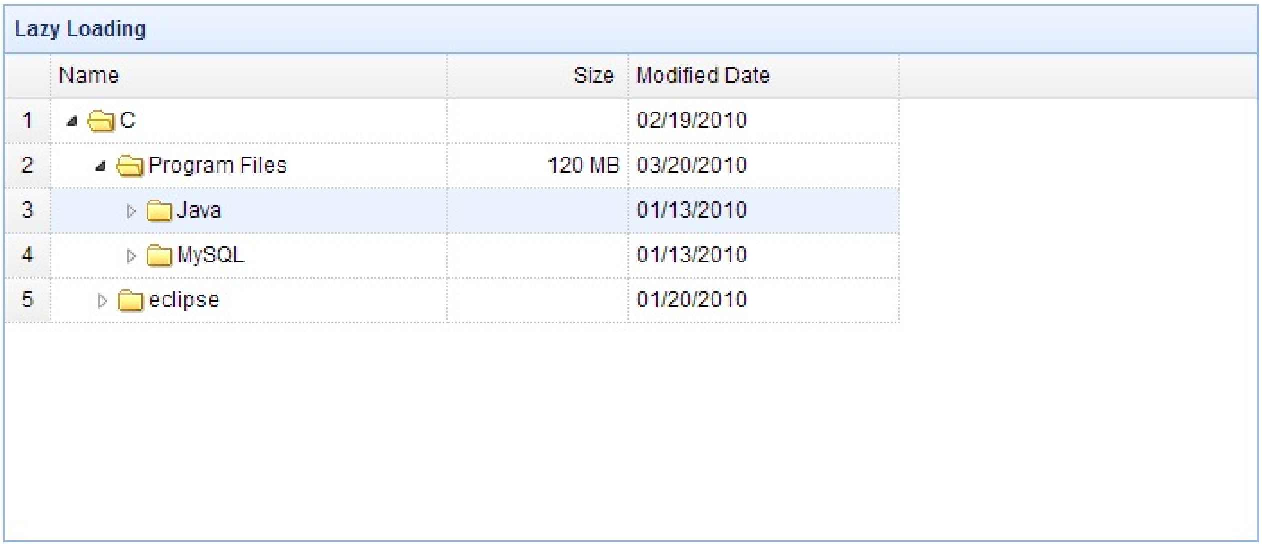
当展开一个行节点时，'id' 值是大于 0 的。 当改变页码时，'id' 值应该被设置为 0 来放置加载子行。

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid4.zip

jQuery EasyUI 树形菜单 - 树形网格惰性加载节点

有时我们已经得到充分的分层树形网格（TreeGrid）的数据。 我们还想让树形网格（TreeGrid）按层次惰性加载节点。 首先，只加载顶层节点。 然后点击节点的展开图标来加载它的子节点。 本教程展示如何创建带有惰性加载特性的树形网格（TreeGrid）。



创建树形网格（TreeGrid）

```
    <table id="test" title="Folder Browser" class="easyui-treegrid'
            data-options="
                url: 'data/treegrid_data.json',
                method: 'get',
                rownumbers: true,
                idField: 'id',
                treeField: 'name',
                loadFilter: myLoadFilter
            ">
        <thead>
            <tr>
                <th field="name" width="220">Name</th>
                <th field="size" width="100" align="right">Size</tl
                <th field="date" width="150">Modified Date</th>
            </tr>
        </thead>
    </table>
```

为了放置加载子节点，我们需要为每个节点重命名 'children' 属性。 正如下面的代码所示，'children' 属性重命名为 'children1'。 当展开一个节点时，我们调用 'append' 方法来加载它的子节点数据。

'loadFilter' 代码

```
    function myLoadFilter(data,parentId){
        function setData(){
            var todo = [];
            for(var i=0; i<data.length; i++){
                todo.push(data[i]);
            }
            while(todo.length){
                var node = todo.shift();
                if (node.children){
                    node.state = 'closed';
                    node.children1 = node.children;
                    node.children = undefined;
                    todo = todo.concat(node.children1);
                }
            }
        }

        setData(data);
        var tg = $(this);
        var opts = tg.treegrid('options');
        opts.onBeforeExpand = function(row){
            if (row.children1){
                tg.treegrid('append',{
                    parent: row[opts.idField],
                    data: row.children1
                });
                row.children1 = undefined;
                tg.treegrid('expand', row[opts.idField]);
            }
            return row.children1 == undefined;
        };
        return data;
    }
```

下载 jQuery EasyUI 实例

jeasyui-tree-treegrid5.zip

jQuery EasyUI 表单

jQuery EasyUI 表单 - 创建异步提交表单

本教程向您展示如何通过 easyui 提交一个表单（Form）。我们创建一个带有 name、email 和 phone 字段的表单。通过使用 easyui 表单（form）插件来改变表单（form）为 ajax 表单（form）。表单（form）提交所有字段到后台服务器，服务器处理和发送一些数据返回到前端页面。我们接收返回数据，并将它显示出来。



创建表单（Form）

```
<div style="padding:3px 2px;border-bottom:1px solid #ccc">Ajax
<form id="ff" action="form1_proc.php" method="post">
    <table>
        <tr>
            <td>Name:</td>
            <td><input name="name" type="text"></input></td>
        </tr>
        <tr>
            <td>Email:</td>
            <td><input name="email" type="text"></input></td>
        </tr>
        <tr>
            <td>Phone:</td>
            <td><input name="phone" type="text"></input></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Submit"></input></t
        </tr>
    </table>
</form>
```

改变为 Ajax 表单

```
    $('#ff').form({
        success:function(data){
            $.messager.alert('Info', data, 'info');
        }
    });
```

服务器端代码

form1_proc.php

```
    $name = $_POST['name'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];

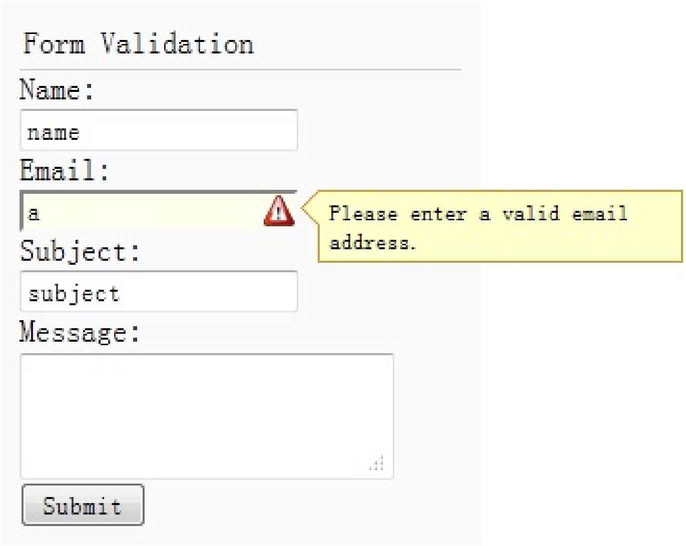
    echo "Your Name: $name <br/> Your Email: $email <br/> Your Phor
```

下载 jQuery EasyUI 实例

jeasyui-form-form1.zip

jQuery EasyUI 表单 - 表单验证

本教程将向您展示如何验证一个表单。easyui 框架提供一个 validatebox 插件来验证一个表单。在本教程中，我们将创建一个联系表单，并应用 validatebox 插件来验证表单。然后您可以根据自己的需求来调整这个表单。



创建表单（form）

让我们创建一个简单的联系表单，带有 name、email、subject 和 message 字段：

```
<div style="padding:3px 2px;border-bottom:1px solid #ccc">Form
<form id="ff" method="post">
    <div>
        <label for="name">Name:</label>
        <input class="easyui-validatebox" type="text" name="nar
    </div>
    <div>
        <label for="email">Email:</label>
        <input class="easyui-validatebox" type="text" name="ema
    </div>
    <div>
        <label for="subject">Subject:</label>
        <input class="easyui-validatebox" type="text" name="sul
    </div>
    <div>
        <label for="message">Message:</label>
        &lt;textarea name="message" style="height:60px;"&gt;&lt
    </div>
    <div>
        <input type="submit" value="Submit">
    </div>
</form>
```

我们添加一个样式名为 easyui-validatebox 到 input 标记，所以 input 标记将根据 validType 属性应用验证。

当表单无效时阻止表单提交

当用户点击表单的 submit 按钮时，如果表单是无效的，我们应该阻止表单提交。

```
$('#ff').form({
    url:'form3_proc.php',
    onSubmit:function(){
        return $(this).form('validate');
    },
    success:function(data){
        $.messager.alert('Info', data, 'info');
    }
});
```

如果表单是无效的，将显示一个提示信息。

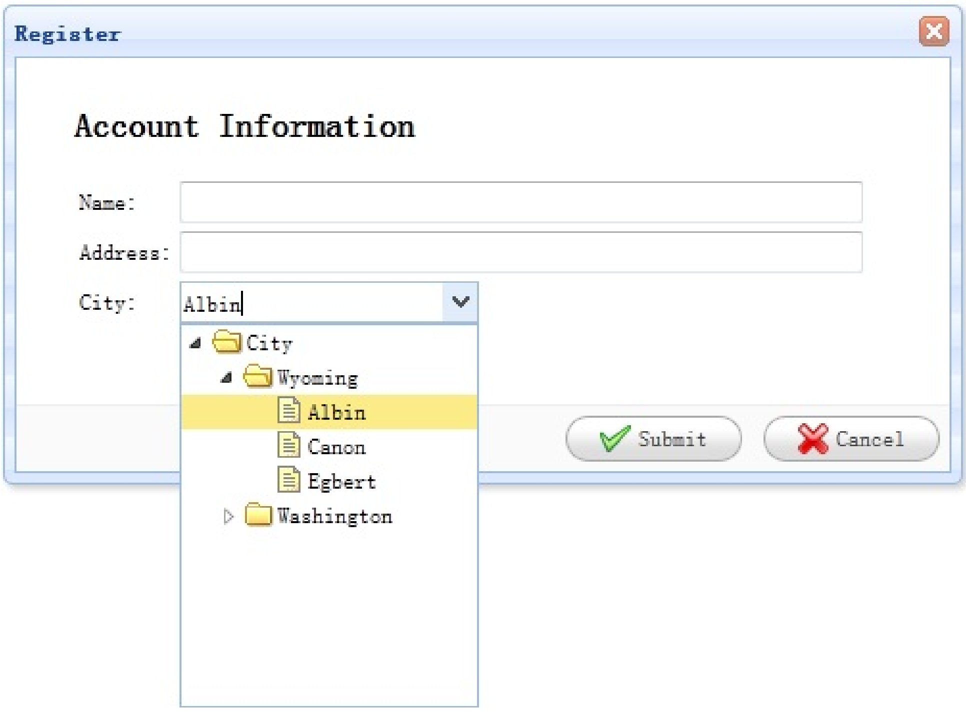
下载 jQuery EasyUI 实例

jeasyui-form-form3.zip

jQuery EasyUI 表单 - 创建树形下拉框

树形下拉框（ComboTree）是一个带有下列树形结构（Tree）的下拉框（ComboBox）。它可以作为一个表单字段进行使用，可以提交给远程服务器。

在本教程中，我们将要创建一个注册表单，带有 name、address、city 字段。city 字段是一个树形下拉框（ComboTree）字段，在里面用户可以下拉树面板（tree panel），并选择一个特定的城市。



创建表单（Form）

```
<div id="dlg" class="easyui-dialog" style="width:500px;height:2
        title="Register" buttons="#dlg-buttons">
    <h2>Account Information</h2>
    <form id="ff" method="post">
        <table>
            <tr>
                <td>Name:</td>
                <td><input type="text" name="name" style="widtl
            </tr>
            <tr>
                <td>Address:</td>
                <td><input type="text" name="address" style="wi
            </tr>
            <tr>
                <td>City:</td>
                <td><select class="easyui-combotree" url="data/
            </tr>
        </table>
    </form>
</div>
<div id="dlg-buttons">
    <a href="#" class="easyui-linkbutton" iconCls="icon-ok" onc
    <a href="#" class="easyui-linkbutton" iconCls="icon-cancel'
</div>
```

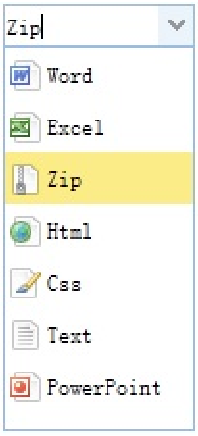
从上面的代码可以看到，我们为一个名为 'city' 的树形下拉框（ComboTree）字段设置了一个 url 属性，这个字段可以从远程服务器检索树形结构（Tree）数据。请注意，这个字段有一个样式名字叫 'easyui-combotree'，所以我们不需要写任何的 js 代码，树形下拉框（ComboTree）字段将自动渲染。

下载 jQuery EasyUI 实例

jeasyui-form-form2.zip

jQuery EasyUI 表单 - 格式化下拉框

本教程向您展示如何创建一个简单的下拉框（Combobox），让它在下拉框中显示图片项。您可以在下拉框（combobox）上使用 formatter 函数来告诉它如何格式化每一个条目。



创建图像下拉框（Combobox）

```
<input id="cc" style="width:100px"
        url="data/combobox_data.json"
        valueField="id" textField="text">
</input>
```

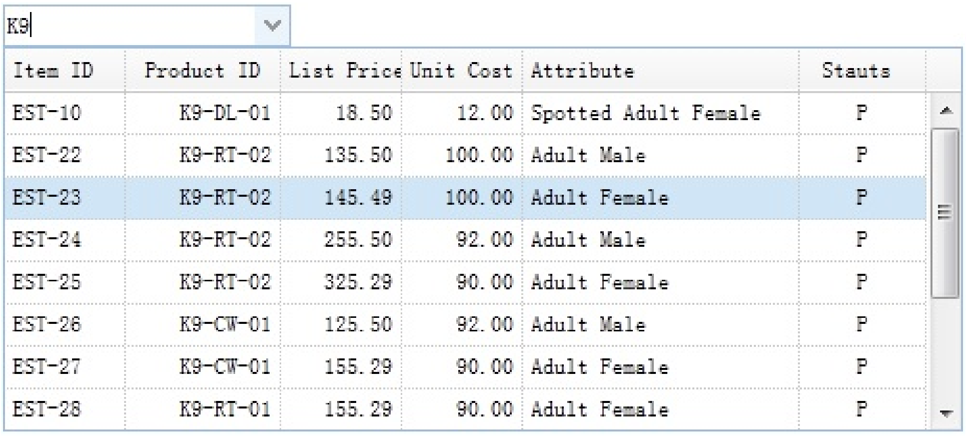
```
$('#cc').combobox({
    formatter:function(row){
        var imageFile = 'images/' + row.icon;
        return '<img class="item-img" src="'+imageFile+'"/><spa
    }
});
```

下载 jQuery EasyUI 实例

jeasyui-form-form4.zip

jQuery EasyUI 表单 - 过滤下拉数据网格

下拉数据网格（Combogrid）组件和下拉框（Combobox）组件的共同点是，除了都具有下拉面板以外，它们都是基于数据网格（Datagrid）的。 下拉数据网格（Combogrid）组件可以过滤、分页，并具有其他一些数据网格（Datagrid）的功能。 本教程向您展示如何在一个下拉数据网格（Combogrid）组件中过滤数据记录。



创建下拉数据网格（Combogrid）

```
<input id="cg" style="width:150px">
```

```
$('#cg').combogrid({
    panelWidth:500,
    url: 'form5_getdata.php',
    idField:'itemid',
    textField:'productid',
    mode:'remote',
    fitColumns:true,
    columns:[[
        {field:'itemid',title:'Item ID',width:60},
        {field:'productid',title:'Product ID',align:'right',width:8
        {field:'listprice',title:'List Price',align:'right',width:6
        {field:'unitcost',title:'Unit Cost',align:'right',width:60
        {field:'attr1',title:'Attribute',width:150},
        {field:'status',title:'Stauts',align:'center',width:60}
    ]]
});
```

下拉数据网格（Combogrid）组件应该定义 'idField' 和 'textField' 属性。 'idField' 属性存储组件值，'textField' 属性在 input 文本框中显示文本消息。 下拉数据网格（Combogrid）组件可以以 'local' 或 'remote' 模式过滤记录。 在远程（remote）模式中，当用户输入字符到 input 文本框中，下拉数据网格（Combogrid）将发送 'q' 参数到远程服务器。

服务器端代码

form5_getdata.php

```
$q = isset($_POST['q']) ? strval($_POST['q']) : '';

include 'conn.php';

$rs = mysql_query("select * from item where itemid like '%$q%' or p
$rows = array();
while($row = mysql_fetch_assoc($rs)){
    $rows[] = $row;
}
echo json_encode($rows);
```

下载 jQuery EasyUI 实例

jeasyui-form-form5.zip

jQuery EasyUI 插件

jQuery EasyUI 提供了用于创建跨浏览器网页的完整的组件集合，包括功能强大的 datagrid（数据网格）、treegrid（树形表格）、 panel（面板）、combo（下拉组合）等等。 用户可以组合使用这些组件，也可以单独使用其中一个。

插件列表

Base（基础）

- Parser 解析器
- Easyloader 加载器
- Draggable 可拖动
- Droppable 可放置
- Resizable 可调整尺寸
- Pagination 分页
- Searchbox 搜索框
- Progressbar 进度条
- Tooltip 提示框

Layout（布局）

- Panel 面板
- Tabs 标签页/选项卡
- Accordion 折叠面板
- Layout 布局

Menu（菜单）与 Button（按钮）

- Menu 菜单
- Linkbutton 链接按钮
- Menubutton 菜单按钮
- Splitbutton 分割按钮

Form（表单）

- Form 表单
- Validatebox 验证框
- Combo 组合
- Combobox 组合框
- Combotree 组合树
- Combogrid 组合网格
- Numberbox 数字框

- Datebox 日期框
- Datetimebox 日期时间框
- Calendar 日历
- Spinner 微调器
- Numberspinner 数值微调器
- Timespinner 时间微调器
- Slider 滑块

Window（窗口）

- Window 窗口
- Dialog 对话框
- Messager 消息框

DataGrid（数据网格）与 Tree（树）

- Datagrid 数据网格
- Propertygrid 属性网格
- Tree 树
- Treegrid 树形网格

插件

easyui 的每个组件都有属性、方法和事件。用户可以很容易地对这些组件进行扩展。

属性

属性是定义在 jQuery.fn.{plugin}.defaults。比如，dialog 的属性是定义在 jQuery.fn.dialog.defaults。

事件

事件（回调函数）也是定义在 jQuery.fn.{plugin}.defaults。

方法

调用方法的语法：$('selector').plugin('method', parameter);

其中：

- selector 是 jquery 对象选择器。
- plugin 是插件名称。
- method 是与插件相匹配的已存在方法。
- parameter 是参数对象，可以是对象、字符串...

方法是定义在 jQuery.fn.{plugin}.methods。每个方法有两个参数：jq 和 param。第一个参数 'jq' 是必需的，引用 jQuery 对象。第二个参数 'param' 引用方法传递的实际参数。比如，要扩展 dialog 组件的方法名为 'mymove' 的方法，代码如下：

```
$.extend($.fn.dialog.methods, {
    mymove: function(jq, newposition){
        return jq.each(function(){
            $(this).dialog('move', newposition);
        });
    }
});
```

现在您可以调用 'mymove' 方法来移动对话框（dialog）到指定的位置：

```
$('#dd').dialog('mymove', {
    left: 200,
    top: 100
});
```

开始使用 jQuery EasyUI

下载库，并在您的页面中引用 EasyUI CSS 和 JavaScript 文件。

```
&lt;link rel="stylesheet" type="text/css" href="easyui/themes/defau
&lt;link rel="stylesheet" type="text/css" href="easyui/themes/icon.
&lt;script type="text/javascript" src="easyui/jquery-1.7.2.min.js"&
&lt;script type="text/javascript" src="easyui/jquery.easyui.min.js'
```

一旦您引用了 EasyUI 必要的文件，您就可以通过标记或者使用 JavaScript 来定义一个 EasyUI 组件。比如，要顶一个带有可折叠功能的面板，您需要编写如下 HTML 代码：

```
&lt;div id="p" class="easyui-panel" style="width:500px;height:200px
    title="My Panel" iconCls="icon-save" collapsible="true"&gt;
    The panel content
&lt;/div&gt;
```

当通过标记创建组件，'data-options' 属性被用来支持自版本 1.3 以来 HTML5 兼容的属性名称。所以您可以如下重写上面的代码：

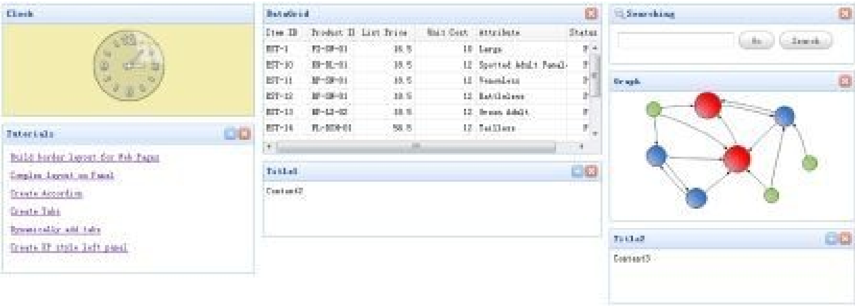
```
&lt;div id="p" class="easyui-panel" style="width:500px;height:200px
    title="My Panel" data-options="iconCls:'icon-save',collapsible
    The panel content
&lt;/div&gt;
```

下面的代码演示了如何创建一个绑定 'onSelect' 事件的组合框。

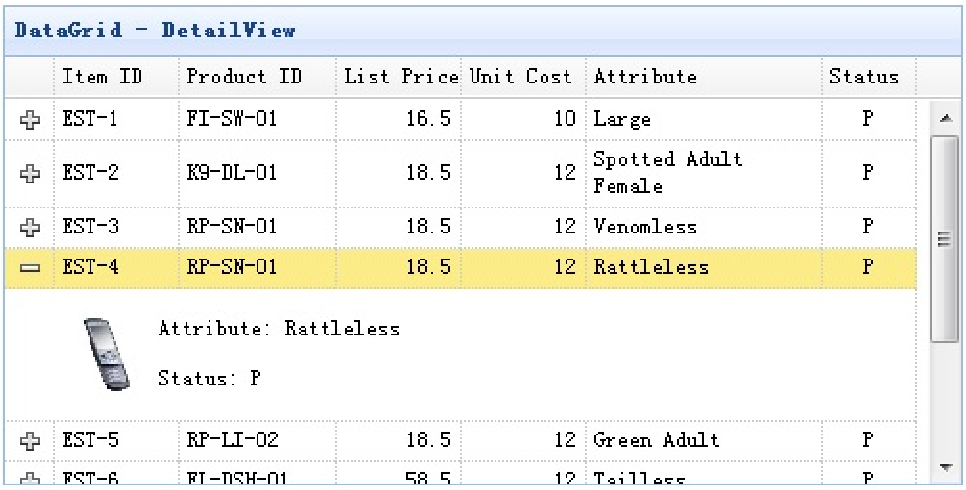
```
&lt;input class="easyui-combobox" name="language"
    data-options="
    url:'combobox_data.json',
    valueField:'id',
    textField:'text',
    panelHeight:'auto',
    onSelect:function(record){
    alert(record.text)
    }"&gt;
```

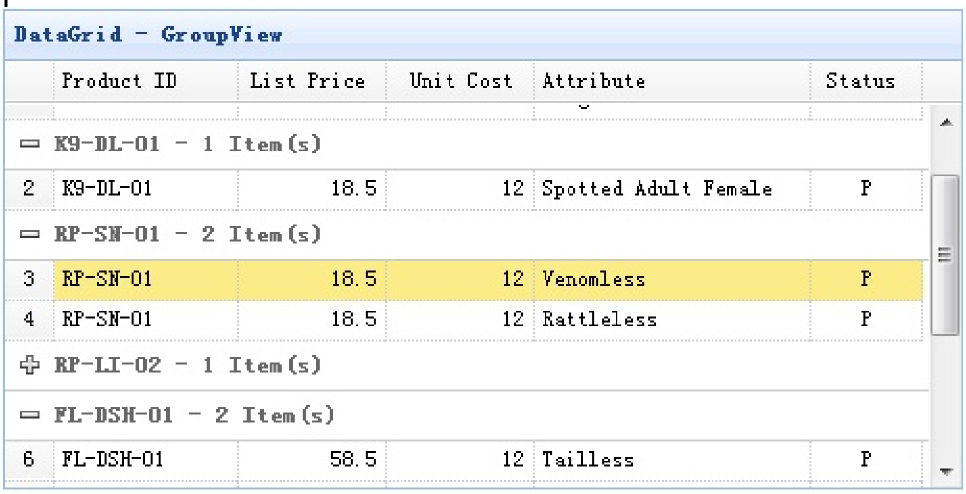
jQuery EasyUI 扩展

Portal（制作图表、列表、球形图等）

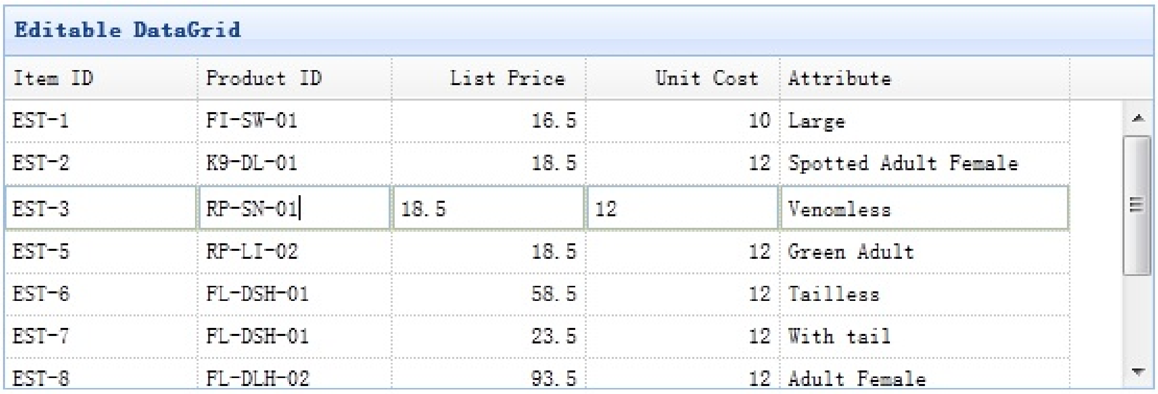


数据网格视图（DataGrid View）

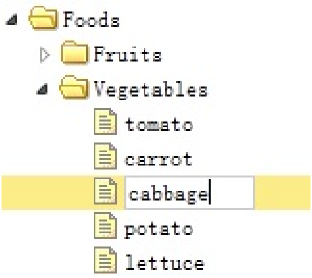




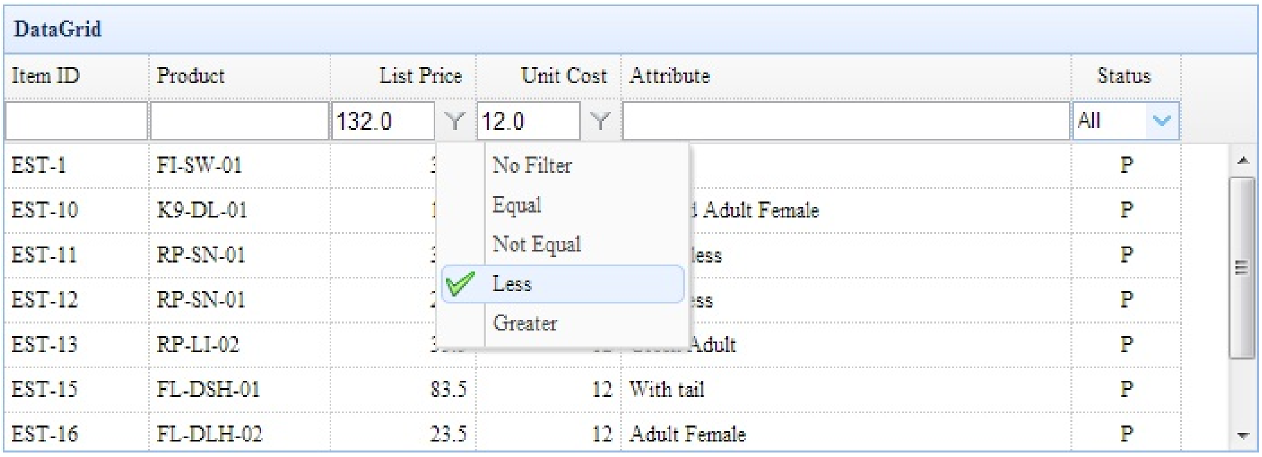
可编辑的数据网格（Editable DataGrid）



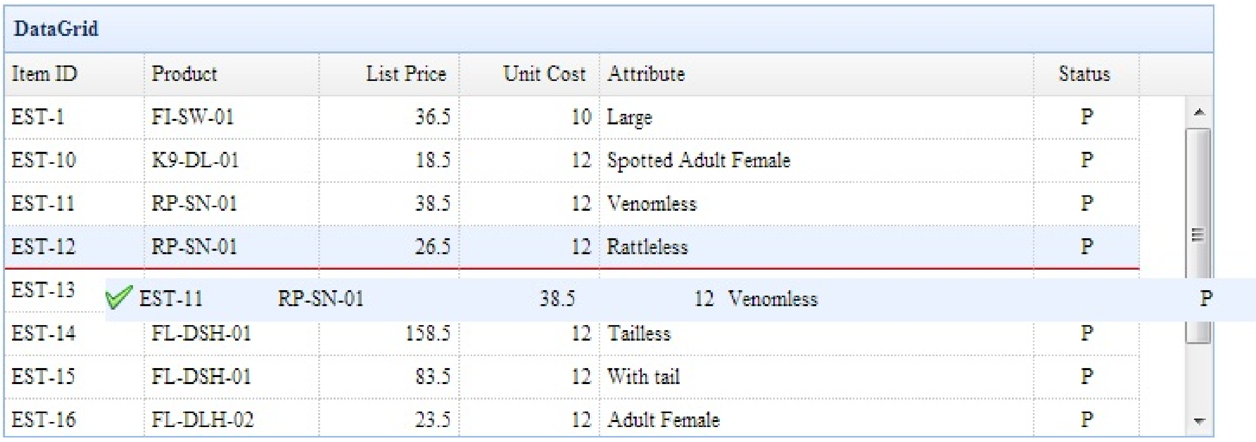
可编辑的树（Editable Tree）



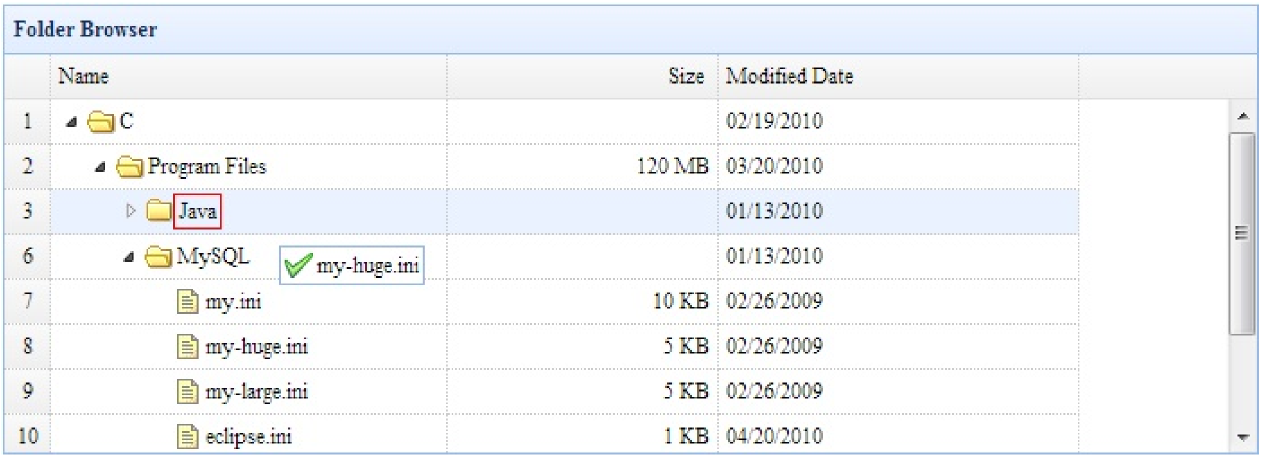
数据网格行过滤（DataGrid Filter Row）



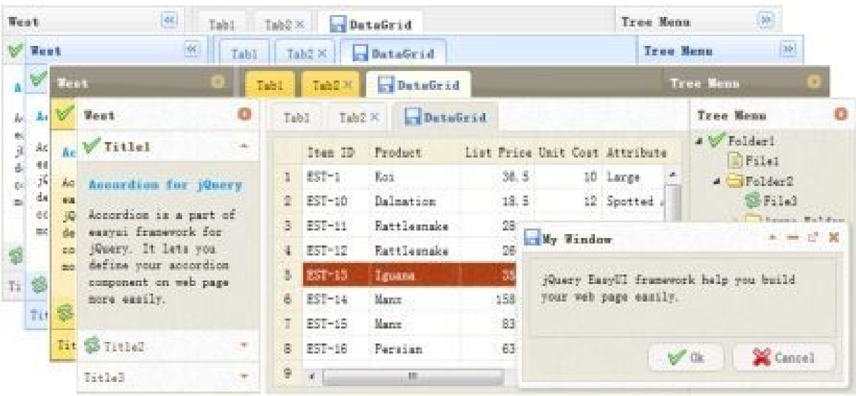
数据网格行拖放（Drag and Drop Rows in DataGrid）



树形网格行拖放（Drag and Drop Rows in TreeGrid）



主题（Themes）

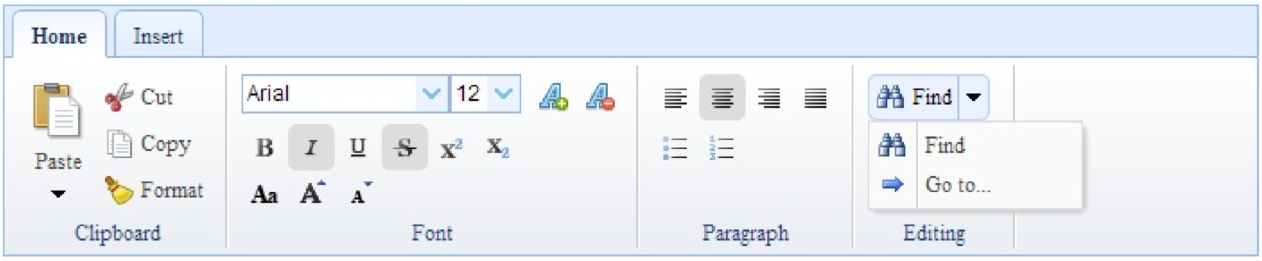


DWR 加载器（DWR Loader）

RTL 支持（RTL support）



Ribbon



W3School Highcharts 教程

作者：W3School

来源：Highcharts 教程

Highcharts 简介

Highcharts 是一个用纯JavaScript编写的一个图表库。

Highcharts 能够很简单便捷的在web网站或是web应用程序添加有交互性的图表

Highcharts 免费提供给个人学习、个人网站和非商业用途使用。

HighCharts 特性

- 兼容性 - 支持所有主流浏览器和移动平台（android、iOS等）。
- 多设备 - 支持多种设备，如手持设备 iPhone/iPad、平板等。
- 免费使用 - 开源免费。
- 轻量 - highcharts.js 内核库大小只有 35KB 左右。
- 配置简单 - 使用 json 格式配置
- 动态 - 可以在图表生成后修改。
- 多维 - 支持多维图表
- 配置提示工具 - 鼠标移动到图表的某一点上有提示信息。
- 时间轴 - 可以精确到毫秒。
- 导出 - 表格可导出为 PDF/ PNG/ JPG / SVG 格式
- 输出 - 网页输出图表。
- 可变焦 - 选中图表部分放大，近距离观察图表；
- 外部数据 - 从服务器载入动态数据。
- 文字旋转 - 支持在任意方向的标签旋转。

支持的图表类型

HighCharts支持的图表类型：

序号	图表类型
1	曲线图
2	区域图
3	饼图
4	散点图
5	气泡图
6	动态图表
7	组合图表
8	**3D** 图
9	测量图
10	热点图
11	树状图**(Treemap)**

接下来几个章节我们讲为大家具体介绍 Highcharts 的使用。

Highcharts 环境配置

本章节我们将为大家介绍如何在网页中使用 Highcharts。

Highcharts 依赖于 jQuery，所以在加载 Highcharts 前必须先加载 jQuery 库。

如果你对 jQuery 不熟悉，可以参阅本站的jQuery 教程。

安装 jQuery

jQuery 安装可以使用以下两种方式：

- 1、访问 jquery.com 下载jQuery包。
- 2、使用百度静态资源库（http://cdn.code.baidu.com/）的jQuery资源：**http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js**

使用下载的方式

使用下载的方式，在 HTML 页面引入 jQuery 代码:

```
<head>  <script  src="/jquery/jquery.min.js"></script>  </head>
```

使用 CDN（推荐）

使用百度静态资源库来加载jQuery库：

```
<head>  <script  src="http://apps.bdimg.com/libs/jquery/2.1.4/jquer
```

安装 Highcharts

Highcharts 安装可以使用以下两种方式：

- 1、访问 highcharts.com 下载 Highcharts 包。
- 2、使用官方提供的 CDN 地址：**http://code.highcharts.com/highcharts.js**

使用下载的方式（推荐）

使用下载的方式，在 HTML 页面引入 Highcharts 代码:

```
<head>  <script  src="/highcharts/highcharts.js"></script>  </head>
```

使用 CDN

使用官方提供的 CDN 地址：

```
<head>  <script  src="http://code.highcharts.com/highcharts.js"></s
```

Highcharts 配置语法

本章节我们将为大家介绍使用 Highcharts 生成图表的一些配置。

第一步：创建 HTML 页面

创建一个 HTML 页面，引入 jQuery 和 Highcharts 库：

文件名：HighchartsTest.htm

```
<html>
<head>
<title>Highcharts 教程 | 菜鸟教程</title>
    <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.mir
   <script src="/try/demo_source/highcharts.js"></script>
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {

});
</script>
</body>
</html>
```

实例中 id 为 container 的 div 用于包含 Highcharts 绘制的图表。

第二步: 创建配置文件

Highcharts 库使用 json 格式来配置。

```
$('#container').highcharts(json);
```

这里 json 表示使用 json 数据格式和 json 格式的配置来绘制图表。步骤如下：

标题

为图表配置标题：

```
var title = {
  text: '月平均气温'
};
```

副标题

为图表配置副标题：

```
var subtitle = {
  text: 'Source: runoob.com'
};
```

X 轴

配置要在 X 轴显示的项。

```
var xAxis = {
   categories: ['一月', '二月', '三月', '四月', '五月', '六月'
      ,'七月', '八月', '九月', '十月', '十一月', '十二月']
};
```

X 轴

配置要在 Y 轴显示的项。

```
var yAxis = {
   title: {
      text: 'Temperature (\xB0C)'
   },
   plotLines: [{
      value: 0,
      width: 1,
      color: '#808080'
   }]
};
```

提示信息

配置提示信息：

```
var tooltip = {
   valueSuffix: '\xB0C'
}
```

展示方式

配置图表向右对齐：

```
var legend = {
   layout: 'vertical',
   align: 'right',
   verticalAlign: 'middle',
   borderWidth: 0
};
```

数据

配置图表要展示的数据。每个系列是个数组，每一项在图片中都会生成一条曲线。

```
var series =  [
   {
      name: 'Tokyo',
      data: [7.0, 6.9, 9.5, 14.5, 18.2, 21.5, 25.2,
         26.5, 23.3, 18.3, 13.9, 9.6]
   },
   {
      name: 'New York',
      data: [-0.2, 0.8, 5.7, 11.3, 17.0, 22.0, 24.8,
         24.1, 20.1, 14.1, 8.6, 2.5]
   },
   {
      name: 'Berlin',
      data: [-0.9, 0.6, 3.5, 8.4, 13.5, 17.0, 18.6,
         17.9, 14.3, 9.0, 3.9, 1.0]
   },
   {
      name: 'London',
      data: [3.9, 4.2, 5.7, 8.5, 11.9, 15.2, 17.0,
         16.6, 14.2, 10.3, 6.6, 4.8]
   }
];
```

第三步: 创建 json 数据

组合是由配置信息:

```
var json = {};

json.title = title;
json.subtitle = subtitle;
json.xAxis = xAxis;
json.yAxis = yAxis;
json.tooltip = tooltip;
json.legend = legend;
json.series = series;
Step 4: Draw the chart
$('#container').highcharts(json);
```

实例

以下为完整的实例(HighchartsTest.htm)：

```
<html>
<head>
<title>Highcharts Tutorial</title>
   <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.
   <script src="/try/demo_source/highcharts.js"></script>
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {
   var title = {
       text: '月平均气温'
   };
   var subtitle = {
        text: 'Source: runoob.com'
   };
   var xAxis = {
       categories: ['一月', '二月', '三月', '四月', '五月', '六月'
              ,'七月', '八月', '九月', '十月', '十一月', '十二月']
   };
   var yAxis = {
      title: {
         text: 'Temperature (\xB0C)'
      },
      plotLines: [{
         value: 0,
         width: 1,
         color: '#808080'
      }]
   };

   var tooltip = {
      valueSuffix: '\xB0C'
   }
```

```
    var legend = {
       layout: 'vertical',
       align: 'right',
       verticalAlign: 'middle',
       borderWidth: 0
    };

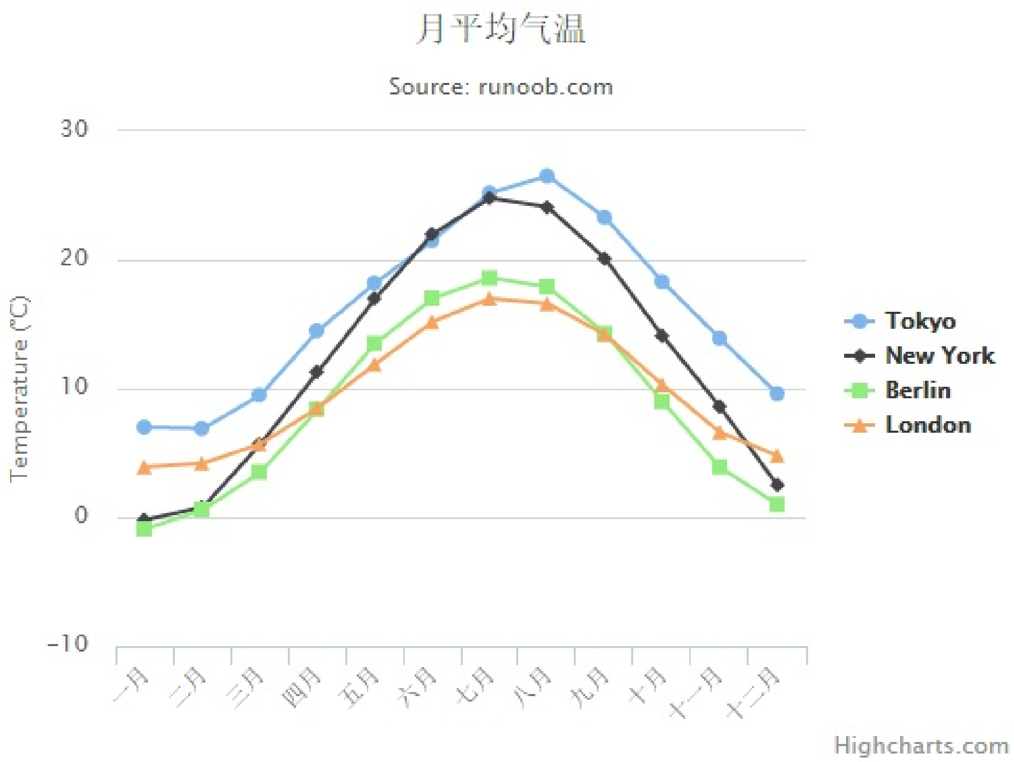
    var series =  [
       {
          name: 'Tokyo',
          data: [7.0, 6.9, 9.5, 14.5, 18.2, 21.5, 25.2,
             26.5, 23.3, 18.3, 13.9, 9.6]
       },
       {
          name: 'New York',
          data: [-0.2, 0.8, 5.7, 11.3, 17.0, 22.0, 24.8,
             24.1, 20.1, 14.1, 8.6, 2.5]
       },
       {
          name: 'Berlin',
          data: [-0.9, 0.6, 3.5, 8.4, 13.5, 17.0, 18.6,
             17.9, 14.3, 9.0, 3.9, 1.0]
       },
       {
          name: 'London',
          data: [3.9, 4.2, 5.7, 8.5, 11.9, 15.2, 17.0,
             16.6, 14.2, 10.3, 6.6, 4.8]
       }
    ];

    var json = {};

    json.title = title;
    json.subtitle = subtitle;
    json.xAxis = xAxis;
    json.yAxis = yAxis;
    json.tooltip = tooltip;
    json.legend = legend;
    json.series = series;

    $('#container').highcharts(json);
});
</script>
</body>
</html>
```

以上实例输出结果为：



月平均气温
Source: runoob.com
30
20
10
0
-10
Temperature (°C)
一月
二月
三月
四月
五月
六月
七月
八月
九月
十月
十一月
十二月
Tokyo
New York
Berlin
London
Highcharts.com

Highcharts 配置选项详细说明

Highcharts 提供大量的配置选项参数，您可以轻松定制符合用户要求的图表，本章节为大家详细介绍Highcharts 配置选项使用说明：

参数配置(属性+事件)

1. chart.events.addSeries：添加数列到图表中。
2. chart.events.click：整个图表的绘图区上所发生的点击事件。
3. chart.events.load：图表加载事件。
4. chart.events.redraw：图表重画事件，当点击图注显示和隐藏绘图时可以触发。
5. chart.events.selection：当图表曲线可选择放大时，当选择图表操作时，可以触发该事件。
6. chart.height：所绘制图表的高度值。
7. chart.inverted：图表中的x，y轴对换。
8. chart.polar：是否为极性图表。
9. chart.reflow：当窗口大小改变时，图表宽度自适应窗口大小改变。
10. chart.renderTo：图表加载的位置，是页面上的一个DOM对象。
11. chart.showAxes：在空白图表中，是否显示坐标轴。
12. chart.type：图表的类型，默认为line，还有bar/column/pie……
13. chart.width：图表绘图区的宽度，默认为自适应。
14. chart.zoomType：图表中数据报表的放大类型，可以以X轴放大，或是以Y轴放大，还可以以XY轴同时放大。
15. colors：图表中多数列时，各数列之间的颜色。是一个数组，一般不动。
16. credits.enabled：是否允许显示版权信息。
17. credits.href：版权所有的链接。
18. credits.text：版权信息显示文字。
19. exporting.buttons.exportButton.enabled：是否允许显示导出按钮。
20. exporting.buttons.exportButton.menuItems：导出按钮的菜单选项。

21. exporting.buttons.exportButton.onclick：导出按钮被点击的事件，不是内部的菜单。
22. exporting.buttons.printButton.enabled：是否允许打印按钮。
23. exporting.buttons.printButton.onclick：打印按钮的点击事件。
24. exporting.enabled：打印和导出按钮是否被允许。
25. exporting.filename：被导出文件的文件名。
26. exporting.type：默认导出图片的文件格式。
27. exporting.url：SVG图表转换并导出的接口处理地址。
28. exporing.width：默认导出图片的宽度。
29. labels：标签，可以加载到图表的任何位置，里面有items，style。
30. lang：语言参数配置，与导出按钮菜单有关的配置，时间名称的配置等。
31. legend.enabled：是否允许图注。
32. navigation.buttonOptions.enabled：图表中所有导航中的按钮是否可被点击。
33. plotOptions.area.allowPointSelect：是否允许数据点的点击。
34. plotOptions.area.color：绘图的颜色。
35. plotOptions.area.dataLabels.enabled：是否允许数据标签。
36. plotOptions.area.enableMouseTracking：是否允许数据图表中，数据点的鼠标跟踪气泡显示。
37. plotOptions.area.events.checkboxClick：数据图表中图注中复选框的点击事件。
38. plotOptions.area.events.click：数据图表中，数据点的点击事件。
39. plotOptions.area.events.hide：数据图表中，某一数据序列隐藏时的事件。
40. plotOptions.area.events.show：数据图表中，某一数据序列显示时的事件。
41. plotOptions.area.events.legendItemClick：数据图表中，图注中的项目被点击时的事件，直接赋值false，则不可点击。
42. plotOptions.area.events.mouseOut：数据点的鼠标移出事件。
43. plotOptions.area.events.mouseOver：数据点的鼠标经过事件。
44. plotOptions.area.marker.enabled：图表中绘图中是否显示点的标记符。
45. plotOptions.area.marker.states.hover.enabled：是否允许标记符的鼠标经过状态。
46. plotOptions.area.marker.states.select.enabled：是否允许标记符的选择状态。

47. plotOptions.area.point.events.click：图表中每一个单独的点点击事件。
48. plotOptions.area.point.events.mouseOut
49. plotOptions.area.point.events..mouseOver
50. plotOptions.area.point.events.remove：删除图表中的点时的事件。
51. plotOptions.area.point.events.select：图表中点选择事件。
52. plotOptions.area.point.events.unselect：图表中点取消选择时的事件。
53. plotOptions.area.point.events.update：图表中数据发生更新时的事件。
54. plotOptions.area.visible：加载时，数据序列默认是显示还是隐藏。
55. plotOptions.area.zIndex：在多序列的情况下，调整每一个序列的层叠顺序。
56. 以上的point.events同样还适用于其他面积类图表（arearange、areaspline、areasplinerange），其他的柱状图（bar、column）及所有图表。
57. plotOptions.area.showInLegend：是否在图注中显示。
58. plotOptions.area.stacking：是以值堆叠，还是以百分比堆叠。
59. plotOptions.area.states.hover.enabled：鼠标放上的状态是否允许。
60. plotOptions.area.stickyTracking：鼠标粘性跟踪数据点。
61. plotOptions.arearange，plotOptions.areaspline，plotOptions.areasplinerange类同于plotOptions.area
62. plotOptions.bar.groupPadding：对于柱状图分组，每个分组之间的间隔。
63. plotOptions.bar.grouping：是否对数据进行分组。
64. plotOptions.bar.minPointLength:：定义当point值为零时，点的最小长度为多少
65. plotOptions.bar.showInLegend：是否在图注中显示。
66. plotOptions.bar.stacking：是以值堆叠，还是以百分比堆叠（normal/percent）。
67. plotOptions.column，plotOptions.columnrange类同于plotOptions.bar
68. plotOptions.line的相关配置类似于plotOptions.area配置。
69. plotOptions.pie.ignoreHiddenPoint：在饼状图中，某一个序列经图注点击隐藏后，整个饼状图是重新以100%分配，还是只在原图基础上隐藏，呈现一个缺口。
70. plotOptions.pie.innerSize：绘制饼状图时，饼状图的圆心预留多大的空白。
71. plotOptions.pie.slicedOffset：与allowPointSelect结合使用，当点被点击时，对应的扇区剥离，这个参数即配置离开的距离。

72. plotOptions.pie的其他常用配置参数类同于plotOptions.area,plotOptions.scatter，plotOptions.series，plotOptions.spline的相关配置类似于plotOptions.area配置。
73. series：是一个数组。
74. series.data.color：某一个数据的颜色。
75. series.data.dataLabels：序列中某一个数据的数据标签。
76. series.data.events类同于plotOptions.area.point.events的相关配置。
77. series.data.marker类同于plotOptions.area.marker的相关配置。
78. series.data.name：配置数据点的名称。
79. series.data.sliced：配置在饼图中，扇区的分离距离大小。
80. series.data.x：点的x值。
81. series.data.y：点的y值。
82. series.name：数据序列的名称。
83. series.stack：堆叠的分组索引。
84. series.type：数据序列的展示类型。
85. series.xAxis，series.yAxis：当使用多坐标轴时，指定某个数列对应哪个坐标轴。
86. subtitle：配置图表的子标题。
87. title：配置图表的标题。
88. tooltip：配置图表中数据的气泡提示。
89. tooltip.valueDecimals：允许的小数点位数。
90. tooltip.percentageDecimals：允许百分比的小数点后位数。
91. xAxis，yAxis配置设置坐标轴
92. allowDecimals：坐标轴上是否允许小数。
93. categories：是一个数组，坐标轴的分类。
94. plotLines：绘制主线。
95. tickColor：刻度颜色。
96. tickInterval：刻度的步进值。
97. labels.rotation：刻度标签旋转度数

Chart：图表区选项

Chart图表区选项用于设置图表区相关属性。

参数	描述	默认值
backgroundColor	设置图表区背景色	#FFFFFF
borderWidth	设置图表边框宽度	0
borderRadius	设置图表边框圆角角度	5
renderTo	图表放置的容器，一般在html中放置一个DIV，获取DIV的id属性值	null
defaultSeriesType	默认图表类型line, spline, area, areaspline, column, bar, pie , scatter	0
width	图表宽度，默认根据图表容器自适应宽度	null
height	图表高度，默认根据图表容器自适应高度	null
margin	设置图表与其他元素之间的间距，数组，如[0,0,0,0]	[null]
plotBackgroundColor	主图表区背景色，即X轴与Y轴围成的区域的背景色	null
plotBorderColor	主图表区边框的颜色，即X轴与Y轴围成的区域的边框颜色	null
plotBorderWidth	主图表区边框的宽度	0
shadow	是否设置阴影，需要设置背景色backgroundColor。	false
reflow	是否自使用图表区域高度和宽度，如果没有设置width和height时，会自适应大小。	true
zoomType	拖动鼠标进行缩放，沿x轴或y轴进行缩放，可以设置为：'x','y','xy'	''
events	事件回调，支持addSeries方法，click方法，load方法，selection方法等的回调函数。	

Color：颜色选项

Color颜色选项用于设置图表的颜色方案。

参数	描述	默认值
color	用于展示图表，折线/柱状/饼状等图的颜色，数组形式。	array

Highcharts已经默认提供了多种颜色方案，当要显示的图形多于颜色种类时，多出的图形会自动从第一种颜色方案开始选取。自定义颜色方案的方法：

```
Highcharts.setOptions({
    colors: ['#058DC7', '#50B432', '#ED561B', '#DDDF00', '#24CBE5',
'#FFF263', '#6AF9C4']
});
```

Title：标题选项

Title标题选项用于设置图表的标题相关属性。

参数	描述	默认值
text	标题文本内容。	Chart title
align	水平对齐方式。	center
verticalAlign	垂直对齐方式。	top
margin	标题与副标题之间或者主图表区间的间距。	15
floating	是否浮动，如果为true，则标题可以偏离主图表区，可配合x,y属性使用。	false
style	设置CSS样式。	{color: '#3E576F',

fontSize: '16px'} |

Subtitle：副标题选项

副标题提供的属性选项与标题title大致相同，可参照上述标题选项，值得一提的是副标题的text选项默认为''，即空的，所以默认情况下副标题不显示。

xAxis：X轴选项

X轴选项用于设置图表X轴相关属性。

参数	描述	默认值
categories	设置X轴分类名称，数组，例如：categories: ['Apples', 'Bananas', 'Oranges']	[]
title	X轴名称，支持text、enabled、align、rotation、style等属性	
labels	设置X轴各分类名称的样式style，格式formatter，角度rotation等。	array
max	X轴最大值(categories为空时)，如果为null，则最大值会根据X轴数据自动匹配一个最大值。	null
min	X轴最小值(categories为空时)，如果为null，则最小值会根据X轴数据自动匹配一个最小值。	array
gridLineColor	网格（竖线）颜色	#C0C0C0
gridLineWidth	网格(竖线)宽度	1
lineColor	基线颜色	#C0D0E0
lineWidth	基线宽度	0

yAxis：Y轴选项

Y轴选项与上述xAxis选项基本一致，请参照上表中的参数设置，不再单独列出。

Series：数据列选项

数据列选项用于设置图表中要展示的数据相关的属性。

参数	描述
data	显示在图表中的数据列，可以为数组或者JSON格式的数据。 如：`data:[0, 5, 3, 5]`， 或`data: [{name: 'Point 1',y: 0}, {name: 'Point 2',y: 5}]`
name	显示数据列的名称。
type	数据列类型，支持 area, areaspline, bar, column, line, pie, scatter or spline

plotOptions：数据点选项

plotOptions用于设置图表中的数据点相关属性。plotOptions根据各种图表类型，其属性设置略微有些差异，现将常用选项列出来。

参数	描述	默认值
enabled	是否在数据点上直接显示数据	false
allowPointSelect	是否允许使用鼠标选中数据点	false
formatter	回调函数，格式化数据显示内容	formatter: function() {return this.y;}

Tooltip：数据点提示框

Tooltip用于设置当鼠标滑向数据点时显示的提示框信息。

参数	描述	默认值
enabled	是否显示提示框	true
backgroundColor	设置提示框的背景色	rgba(255, 255, 255, .85)
borderColor	提示框边框颜色，默认自动匹配数据列的颜色	auto
borderRadius	提示框圆角度	5
shadow	是否显示提示框阴影	true
style	设置提示框内容样式，如字体颜色等	color:'#333'
formatter	回调函数，用于格式化输出提示框的显示内容。返回的内容支持html标签如：<b>, <strong>, <i>, <em>, , <span>	2

Legend：图例选项

legend用于设置图例相关属性。

参数	描述	默认值
layout	显示形式，支持水平horizontal和垂直vertical	horizontal
align	对齐方式。	center
backgroundColor	图例背景色。	null
borderColor	图例边框颜色。	#909090
borderRadius	图例边框角度	5
enabled	是否显示图例	true
floating	是否可以浮动，配合x，y属性。	false
shadow	是否显示阴影	false
style	设置图例内容样式	''

更多详细信息请参照highcharts官网英文文档：http://api.highcharts.com/highcharts

Highcharts 曲线图

Highcharts 基本曲线图

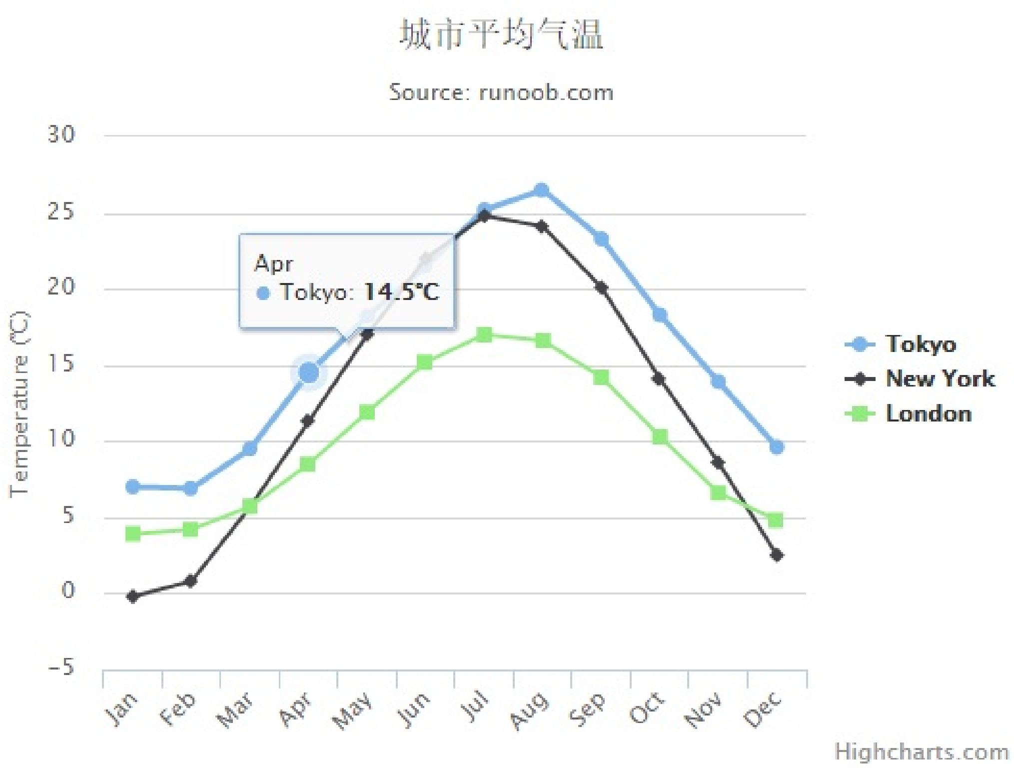
以下实例是基础曲线图。我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看个完整实例：

实例

文件名：highcharts_line_basic.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 带有数据标签曲线图表

以下实例是带有数据标签曲线图表。我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看个完整实例：

实例

文件名：highcharts_line_labels.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 异步加载数据曲线图表

以下实例演示了异步加载数据曲线图表。这边我们通过 jQuery.getJSON() 方法从异步加载 csv 文件：

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看个完整实例：

导入 data.js 文件

异步加载数据需要引入以下js 文件：

```
<script src="http://code.highcharts.com/modules/data.js"></script>
```

配置

X 轴

以每周为间隔设置 X 轴：

```
var xAxis = { tickInterval: 7 * 24 * 3600 * 1000, // 一周
```

Y 轴

以每周为间隔设置 Y 轴：

配置两个 Y 轴：

```
var yAxis = [{ // 左边 Y 轴 title: { text: null }, labels: { a
```

plotOptions

plotOptions用于设置图表中的数据点相关属性。

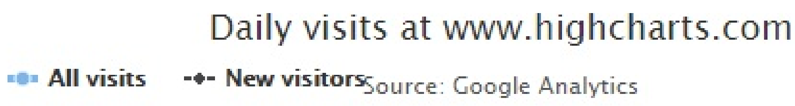
```
 var plotOptions = { series: { cursor: 'pointer', point: { eve
```

实例

文件名：highcharts_line_ajax.html

```
<html>  <head>  <title>Highcharts Tutorial</title>  <title>Highchar
```

以上实例输出结果为：



Highcharts 时间序列，可缩放的图表

以下实例演示了基于时间的曲线图表。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看个完整实例：

配置

图表

配置可缩放图表。 chart.zoomType 指定了用户可以拖放的尺寸，用户可以通过拖动鼠标来放大，可能值是x，y或xy：

```
var chart = { zoomType: 'x' };
```

plotOptions

使用 plotOptions 配置图表区域：

配置两个 Y 轴：

```
var plotOptions = { area: { fillColor: { linearGradient: { x1:
```

plotOptions

plotOptions用于设置图表中的数据点相关属性。

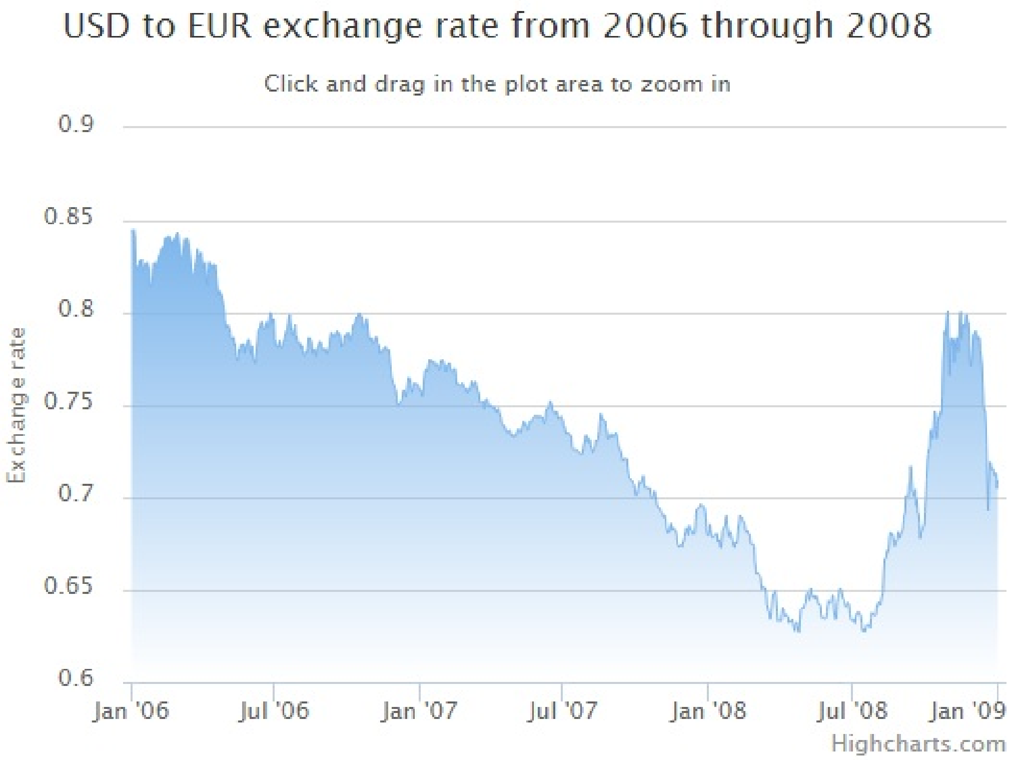
```
  var plotOptions = { series: { cursor: 'pointer', point: { eve
```

实例

文件名：highcharts_line_time.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


USD to EUR exchange rate from 2006 through 2008
Click and drag in the plot area to zoom in
Exchange rate
0.9
0.85
0.8
0.75
0.7
0.65
0.6
Jan '06
Jul '06
Jan '07
Jul '07
Jan '08
Jul '08
Jan '09
Highcharts.com

Highcharts X 轴翻转曲线图

以下实例演示了 X 轴翻转曲线图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看个完整实例：

配置

配置图表类型 type 为 spline。chart.type 默认为 "line"。

配置 X 轴翻转。inverted 设置为 true 即 X 轴翻转，默认为 false。

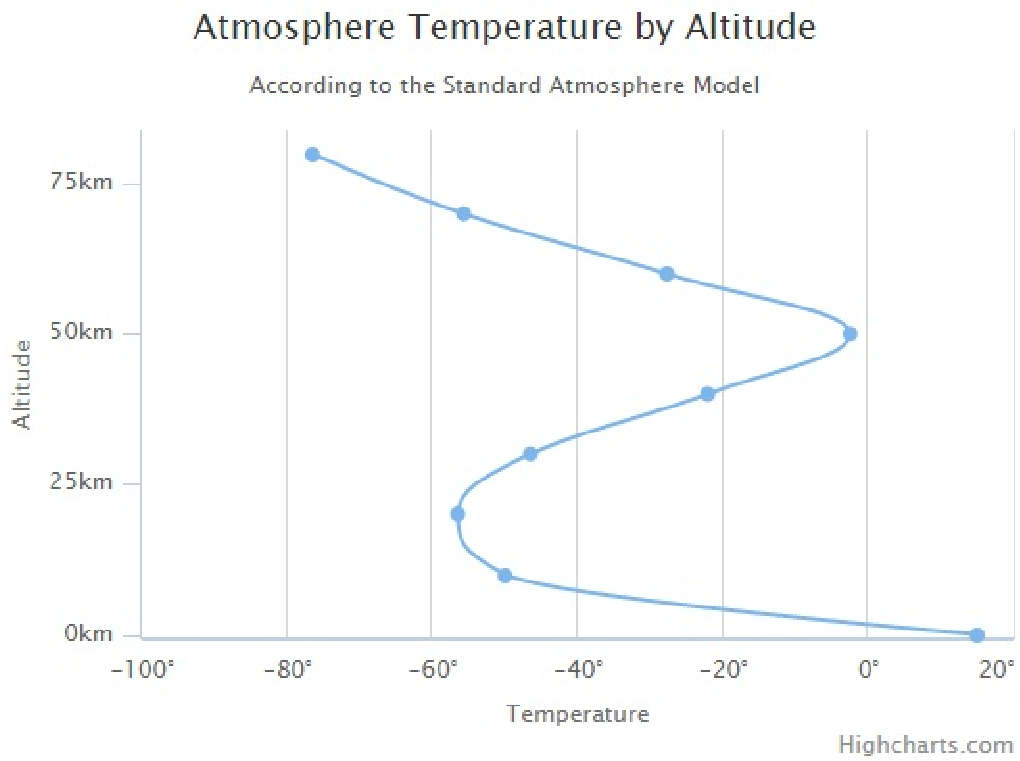
```
chart var chart =  { type:  'spline', inverted:  true  };
```

实例

文件名：highcharts_spline_inverted.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Atmosphere Temperature by Altitude
According to the Standard Atmosphere Model
Altitude
75km
50km
25km
0km
-100°
-80°
-60°
-40°
-20°
0°
20°
Temperature
Highcharts.com

Highcharts 带标记曲线图

以下实例演示了 带标记曲线图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下如何配置。

配置

我们使用 marker.symbol 属性来配置标记。标记可以是 'square', 'diamond' 或 图片 url。标记可以添加在任何的数据点上：

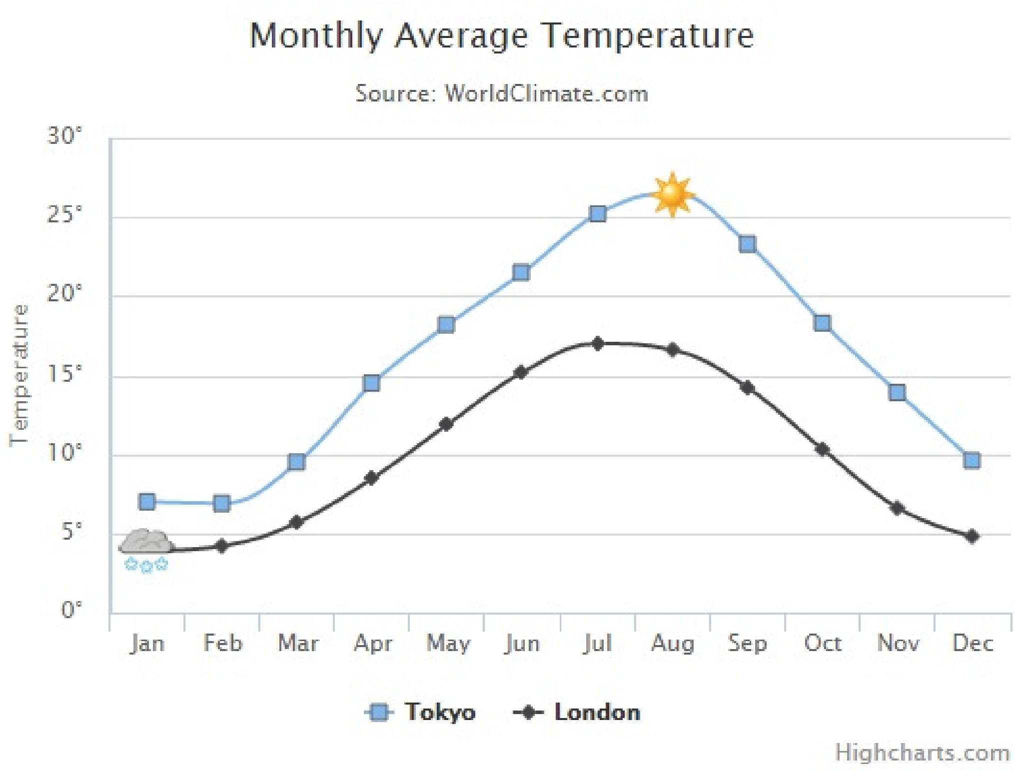
```
var series= [{ name: 'Tokyo', marker: { symbol: 'square' }, da
```

实例

文件名：highcharts_spline_symbols.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 标示区曲线图

以下实例演示了标示区曲线图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下如何配置。

配置

使用 yAxis.plotBands 属性来配置标示区。区间范围使用 'from' 和 'to' 属性。颜色设置使用 'color' 属性。标签样式使用 'label' 属性。

配置信息：

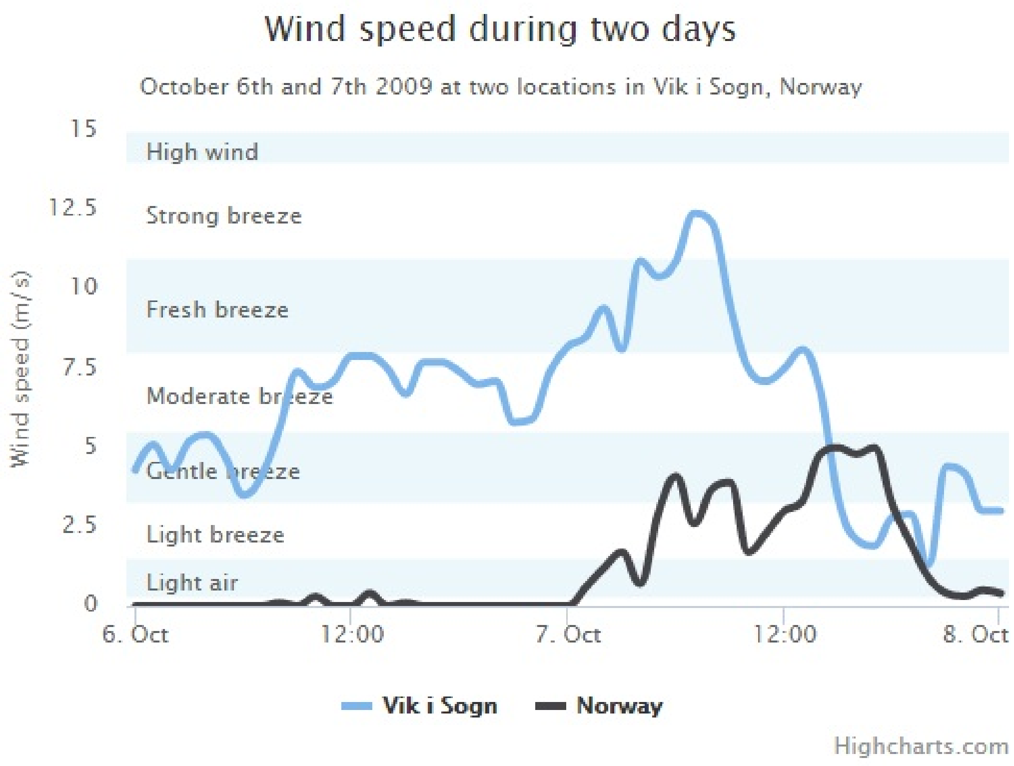
```
var yAxis = { title: { text: 'Wind speed (m/s)' }, min: 0, mir
```

实例

文件名：highcharts_spline_bands.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Wind speed during two days
October 6th and 7th 2009 at two locations in Vik i Sogn, Norway
Wind speed (m/s)
15
12.5
10
7.5
5
2.5
0
High wind
Strong breeze
Fresh breeze
Moderate breeze
Gentle breeze
Light breeze
Light air
6. Oct
12:00
7. Oct
12:00
8. Oct
Vik i Sogn
Norway
Highcharts.com

Highcharts 时间间隔图表

以下实例演示了时间间隔图表。

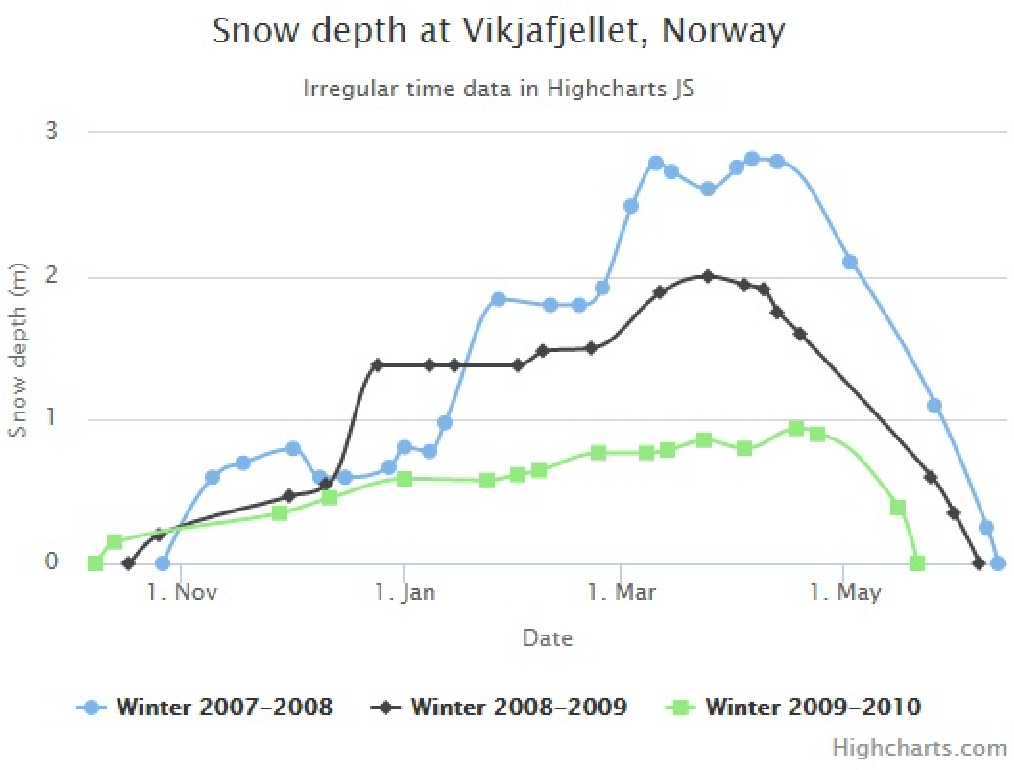
我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下如何配置。

实例

文件名：highcharts_spline_time.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 对数图表

以下实例演示了对数图表。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下如何配置。

配置

配置 yAxis.type 为 'logarithmic'。它定义了 x 轴类型。可选值有 "linear", "logarithmic", "datetime" 或 "category"。默认值为linear。

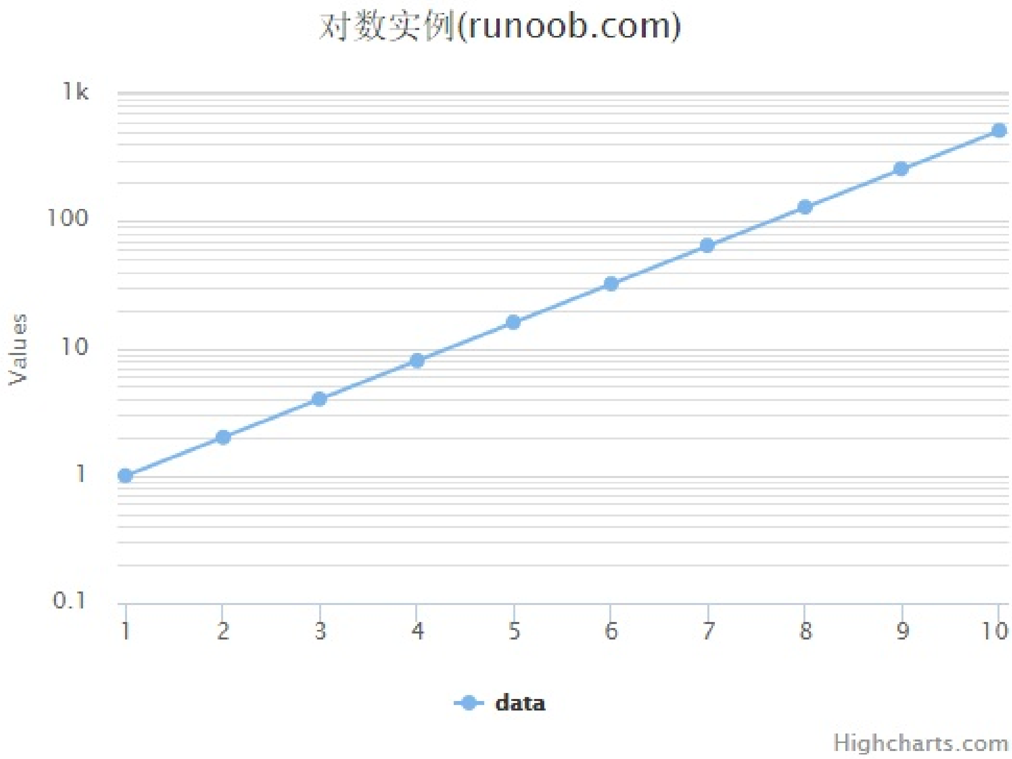
```
yAxis var yAxis =  { type:  'logarithmic', minorTickInterval:  0.1
```

实例

文件名：highcharts_line_logarithmic.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


对数实例(runoob.com)
1k
100
10
1
0.1
Values
1
2
3
4
5
6
7
8
9
10
data
Highcharts.com

Highcharts 区域图

Highcharts 基本区域图

以下实例演示了基础区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置：

配置

chart

chart.type 配置项用于设定图表类型，默认为 "line"，本章节我们使用 'area'。

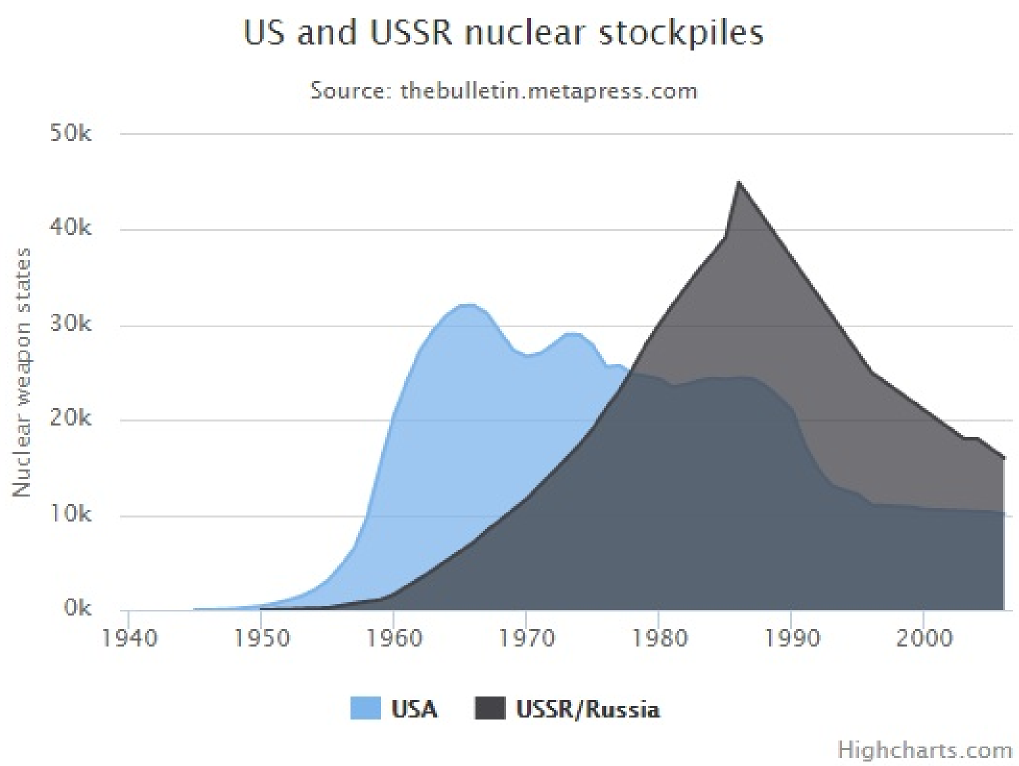
```
var chart = { type: 'area' };
```

实例

文件名：highcharts_area_basic.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title
```

以上实例输出结果为：


US and USSR nuclear stockpiles
Source: thebulletin.metapress.com
Nuclear weapon states
50k
40k
30k
20k
10k
0k
1940
1950
1960
1970
1980
1990
2000
USA
USSR/Russia
Highcharts.com

Highcharts 使用负数区域图

以下实例演示了使用负数区域图。

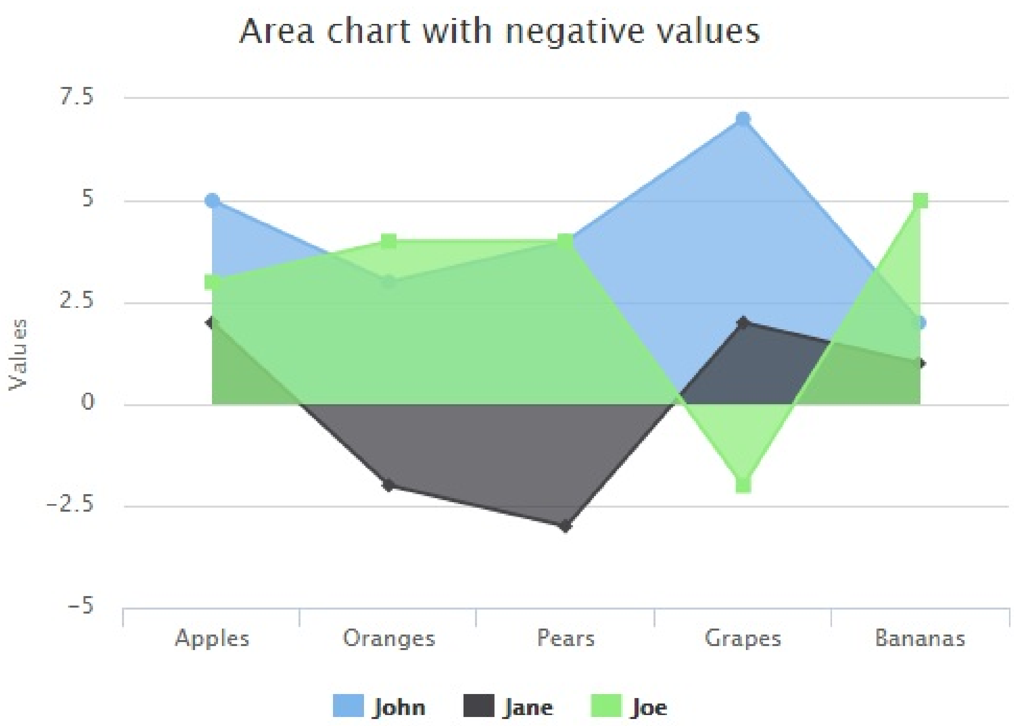
我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的实例：

实例

文件名：highcharts_area_negative.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 堆叠区域图

以下实例演示了堆叠区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 plotOptions 中添加 stacking 属性：

配置

plotOptions：数据点选项

plotOptions用于设置图表中的数据点相关属性。plotOptions根据各种图表类型，其属性设置略微有些差异。

配置图表堆叠设置 plotOptions.area.stacking 为 "normal"。如果禁用堆叠使用 null。 如果值为 "percent" 堆叠则按百分比。

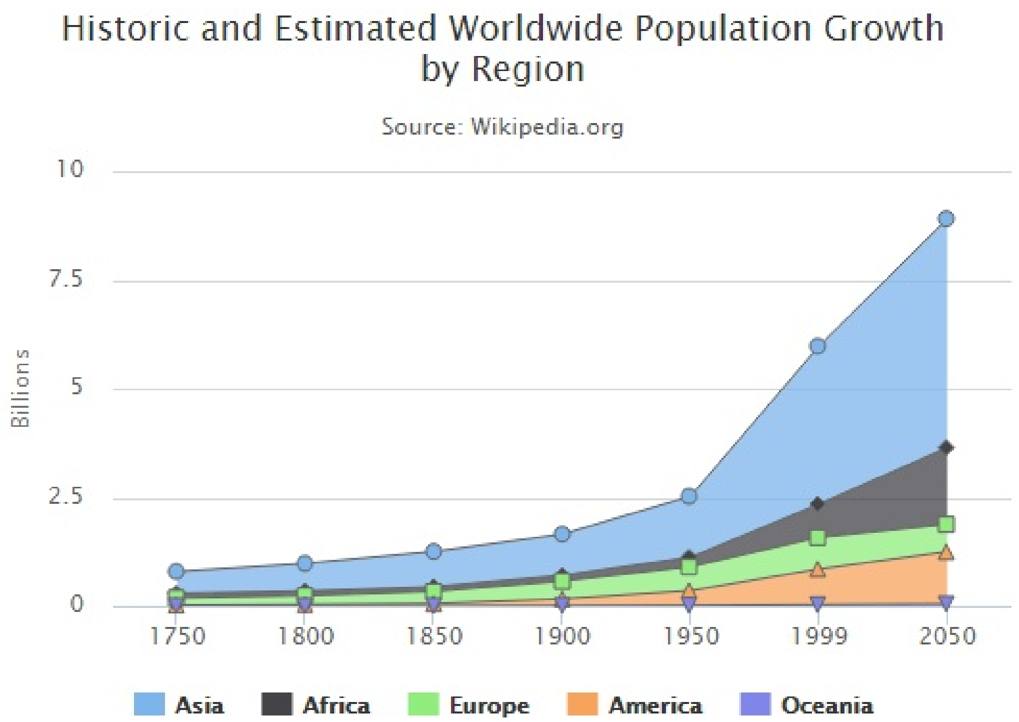
```
var plotOptions =  { area:  { stacking:  'normal', lineColor:  '#66
```

实例

文件名：highcharts_area_stacked.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Historic and Estimated Worldwide Population Growth by Region
Source: Wikipedia.org
Billions
10
7.5
5
2.5
0
1750
1800
1850
1900
1950
1999
2050
Asia
Africa
Europe
America
Oceania

Highcharts 百分比堆叠区域图

以下实例演示了百分比堆叠区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 plotOptions 中添加 stacking 属性：

配置

plotOptions：数据点选项

plotOptions用于设置图表中的数据点相关属性。plotOptions根据各种图表类型，其属性设置略微有些差异。

配置图表堆叠设置 plotOptions.area.stacking 为 "percent"。如果禁用堆叠使用 null。

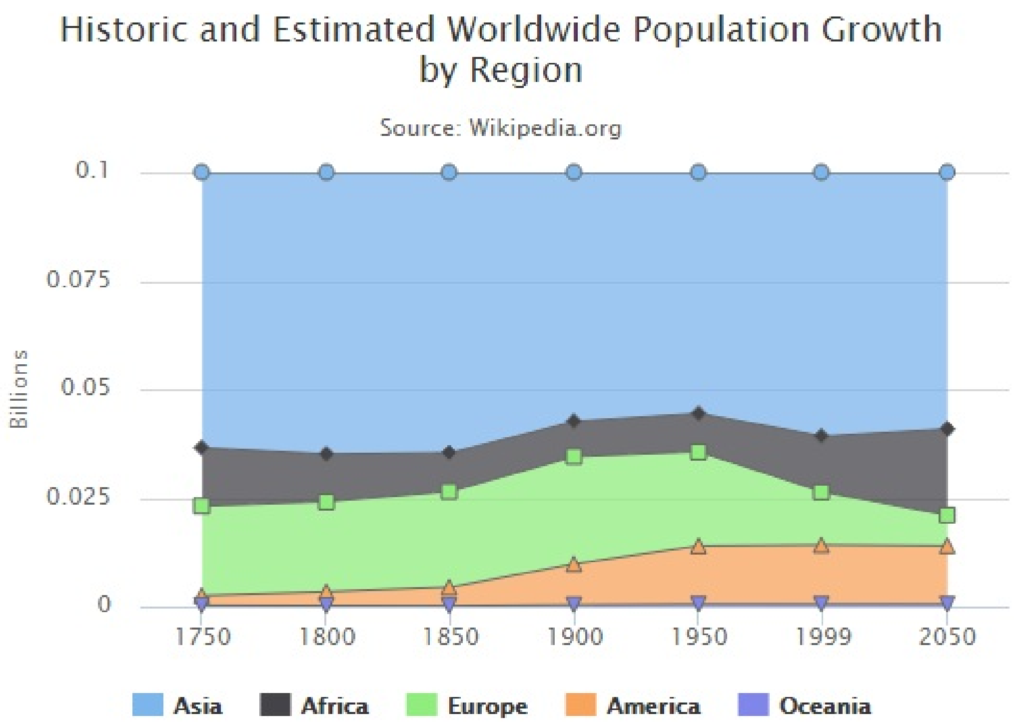
```
var plotOptions = { area: { stacking: 'percent', lineColor: '#6
```

实例

文件名：highcharts_area_percentage.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Historic and Estimated Worldwide Population Growth by Region
Source: Wikipedia.org
Billions
0.1
0.075
0.05
0.025
0
1750
1800
1850
1900
1950
1999
2050
Asia
Africa
Europe
America
Oceania

Highcharts 丢失值区域图

以下实例演示了丢失值区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 chart 中添加 spacingBottom 属性。

chart 配置

将 chart 的 spacingBottom 属性设置为 30。表示图表间的间隔区间。

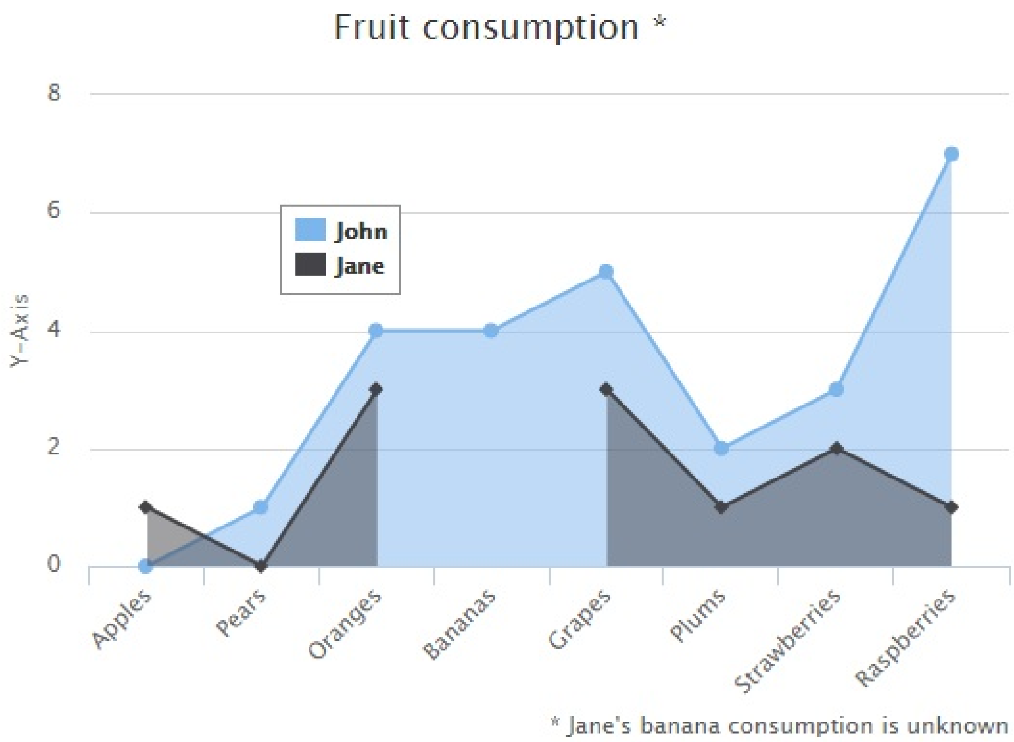
```
var chart = { type:  'area', spacingBottom:  30  };
```

实例

文件名：highcharts_area_missing.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 反转x轴与y轴

以下实例演示了反转x轴与y轴区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 chart 中添加 inverted 属性。

chart 配置

将 chart 的 inverted 属性设置为 true，X轴为垂直，Y轴为水平的。

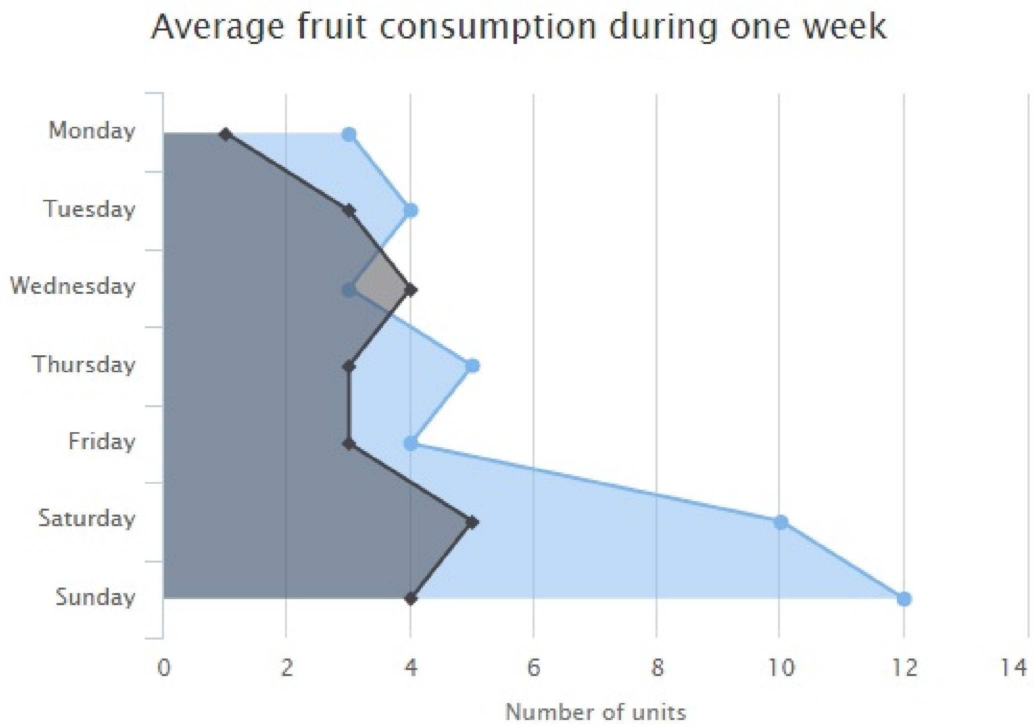
```
var chart = { type: 'area', inverted: true };
```

实例

文件名：highcharts_area_inverted.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 曲线区域图

以下实例演示了曲线区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 chart 中修改 type 属性。

chart 配置

将 chart 的 type 属性设置为 areaspline，chart.type 描述了图表类型。默认值为 "line"。

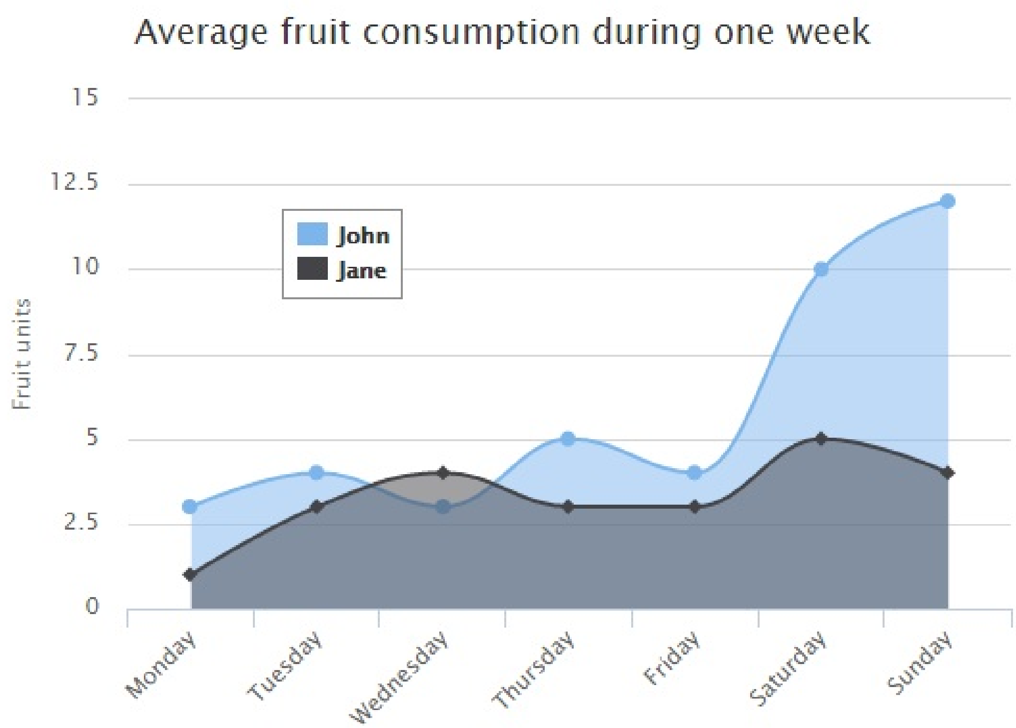
```
var chart = { type: 'areaspline' };
```

实例

文件名：highcharts_area_spline.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 区间区域图

以下实例演示了区间区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 chart 中修改 type 属性。

chart 配置

将 chart 的 type 属性设置为 arearange，chart.type 描述了图表类型。默认值为 "line"。

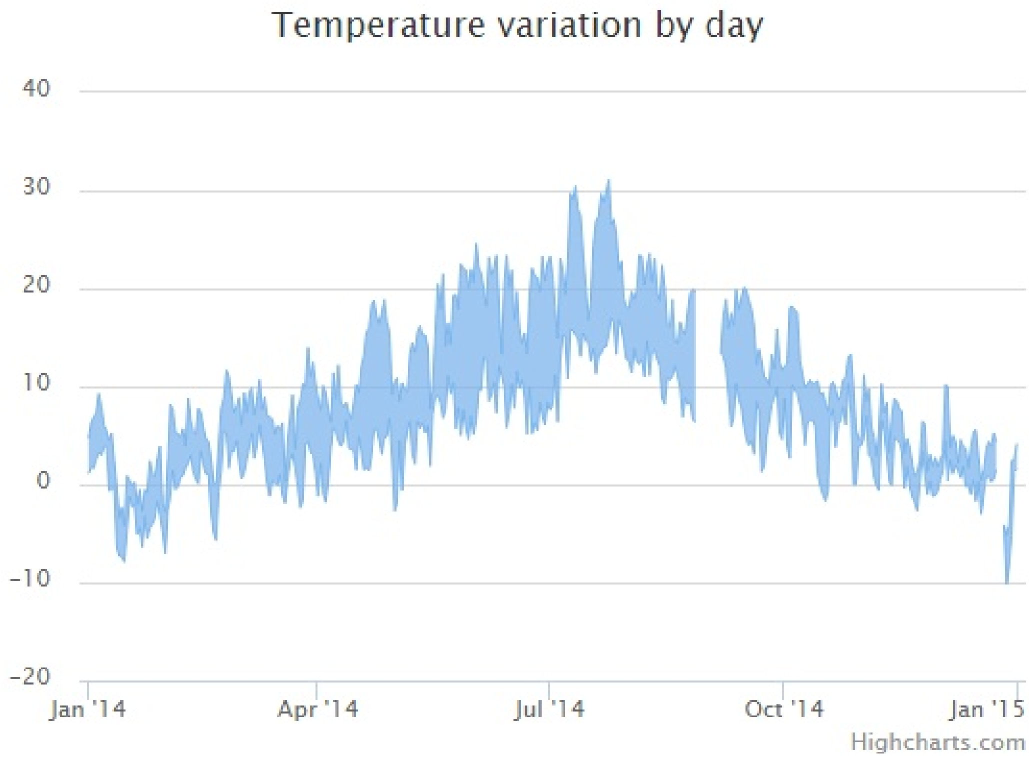
```
var chart = { type: 'arearange' };
```

实例

文件名：highcharts_area_range.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 使用区间和线的区域图

以下实例演示了使用区间和线的区域图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 chart 中修改 type 属性。

chart 配置

将 chart 的 type 属性设置为 arearange，chart.type 描述了图表类型。默认值为 "line"。

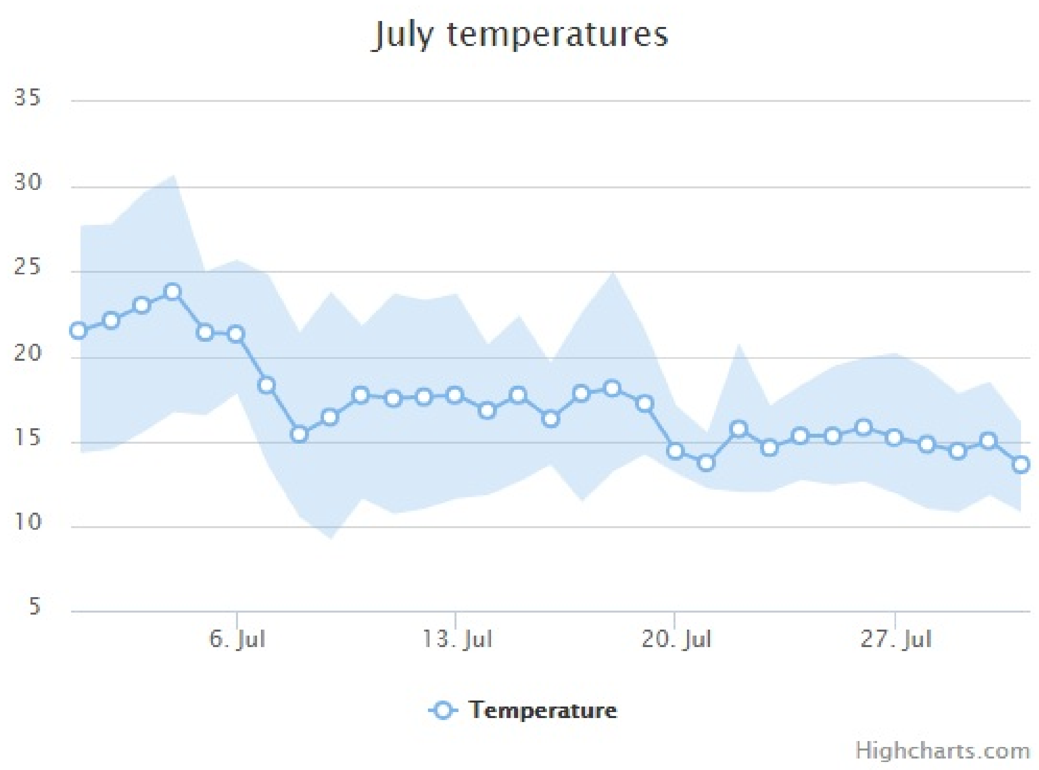
```
var chart = { type: 'arearange' };
```

实例

文件名：highcharts_area_rangeline.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 条形图

Highcharts 基本条形图

以下实例演示了基本条形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart 配置

设置 chart 的 type 属性 为 bar，chart.type 描述了图表类型。默认值为 "line"。

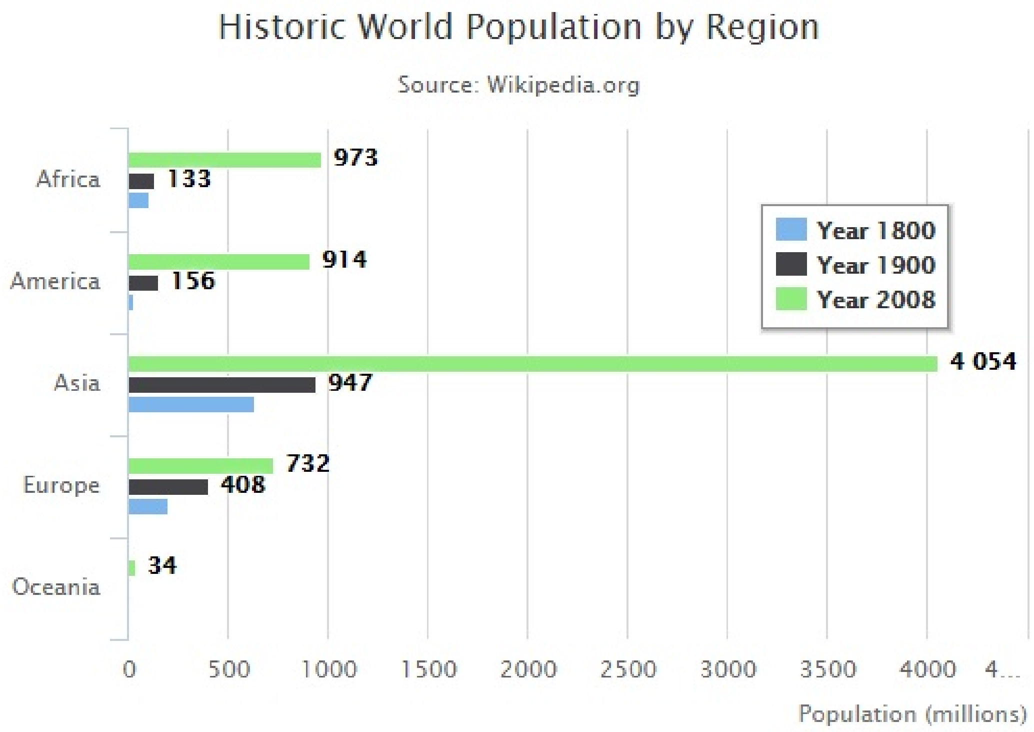
```
var chart = { type: 'bar' };
```

实例

文件名：highcharts_bar_basic.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Historic World Population by Region
Source: Wikipedia.org
Africa
America
Asia
Europe
Oceania
973
133
914
156
4 054
947
732
408
34
Year 1800
Year 1900
Year 2008
0
500
1000
1500
2000
2500
3000
3500
4000
4...
Population (millions)

Highcharts 堆叠条形图

以下实例演示了堆叠条形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

plotOptions

配置图表堆叠使用 plotOptions.series.stacking，并设置为 "normal"。如果禁用堆叠可设置为 null ， "normal" 通过值设置堆叠， "percent" 堆叠则按百分比。

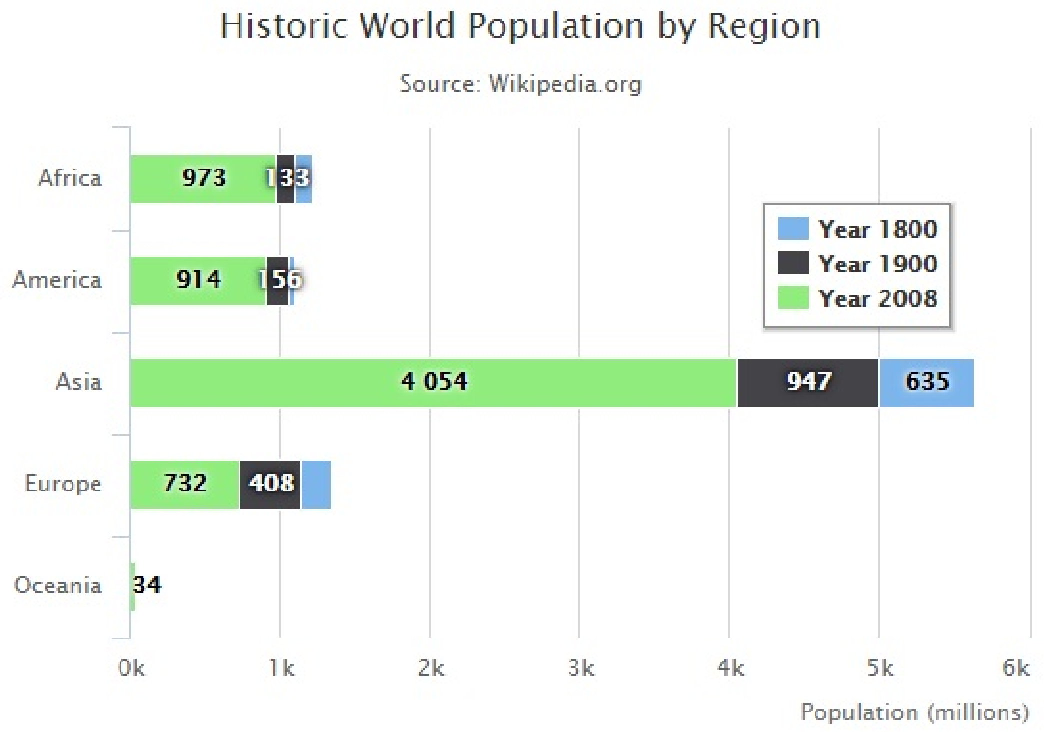
```
var plotOptions = { series: { stacking: 'normal' } };
```

实例

文件名：highcharts_bar_stacked.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Historic World Population by Region
Source: Wikipedia.org
Africa
America
Asia
Europe
Oceania
973
133
914
156
4 054
947
635
732
408
34
Year 1800
Year 1900
Year 2008
0k
1k
2k
3k
4k
5k
6k
Population (millions)

Highcharts 反向条形图

以下实例演示了使用负值的，反向条形图。

实例

文件名：highcharts_bar_negative.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

Highcharts 柱形图

Highcharts 基本柱形图

以下实例演示了基本柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart 配置

设置 chart 的 type 属性 为 column , chart.type 描述了图表类型。默认值为 "line"。

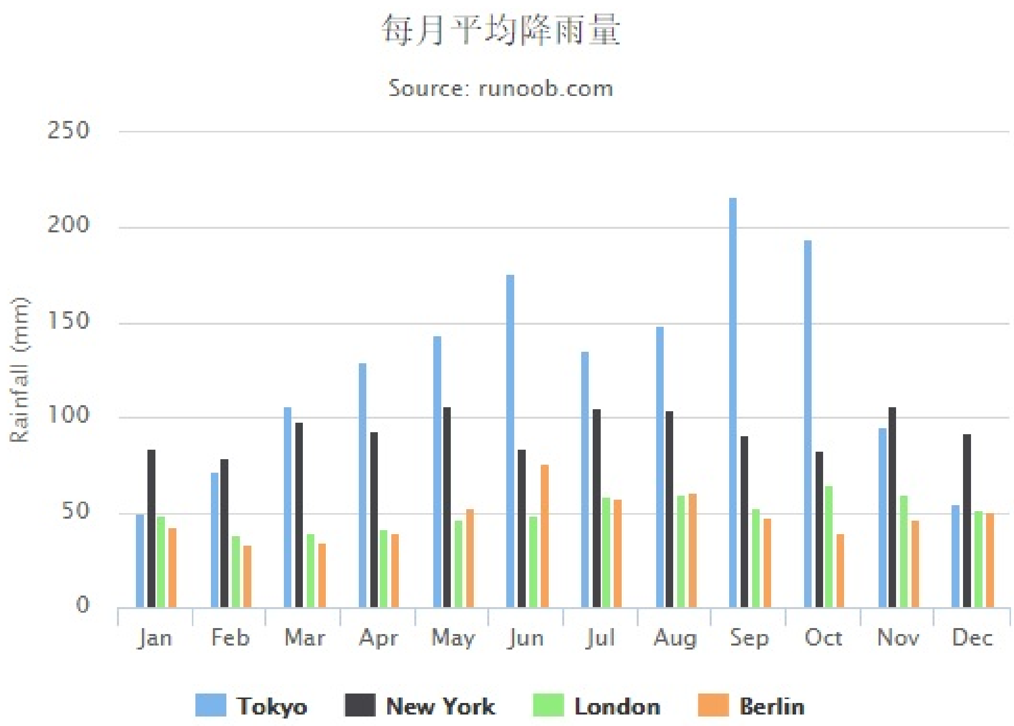
```
var chart =  { type:  'column'  };
```

实例

文件名：highcharts_column_basic.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


每月平均降雨量
Source: runoob.com
250
200
150
100
50
0
Rainfall (mm)
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Tokyo
New York
London
Berlin

Highcharts 使用负值的反向柱形图

以下实例演示了使用负值的反向柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart 配置

设置 chart 的 type 属性 为 column ， chart.type 描述了图表类型。默认值为 "line"。

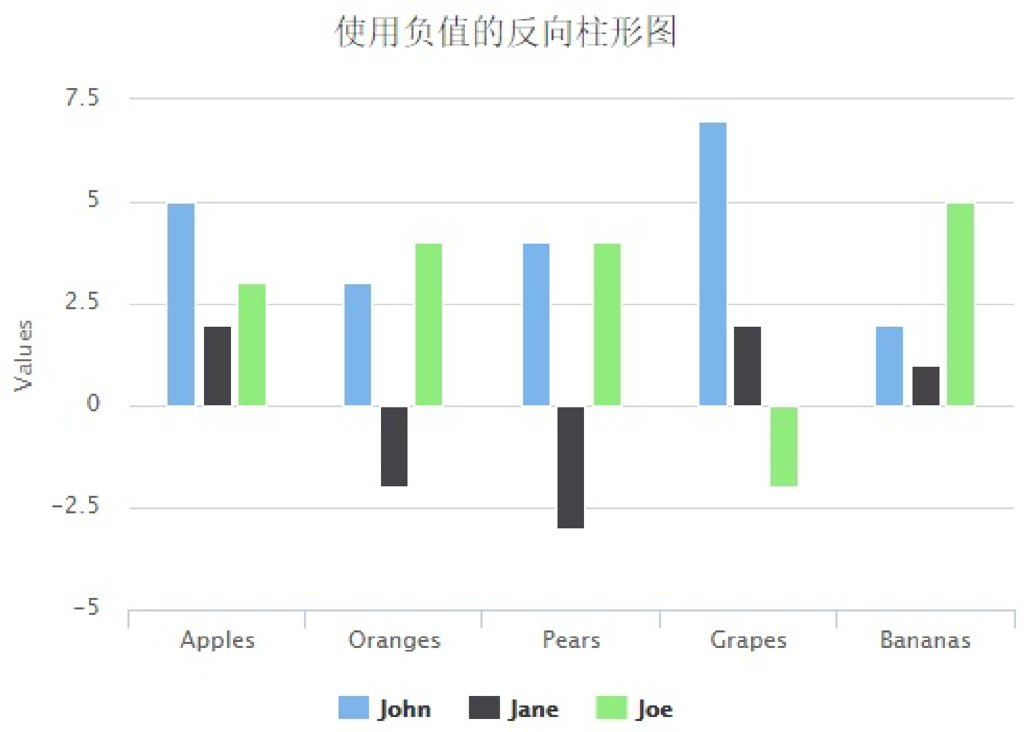
```
var chart = { type: 'column' };
```

实例

文件名：highcharts_column_negative.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



使用负值的反向柱形图
7.5
5
2.5
0
-2.5
-5
Values
Apples
Oranges
Pears
Grapes
Bananas
John
Jane
Joe

Highcharts 堆叠柱形图

以下实例演示了堆叠柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 plotOptions 中添加 stacking 属性：

配置

plotOptions：数据点选项

plotOptions用于设置图表中的数据点相关属性。plotOptions根据各种图表类型，其属性设置略微有些差异。

配置图表堆叠设置 plotOptions.area.stacking 为 "percent"。如果禁用堆叠使用 null。

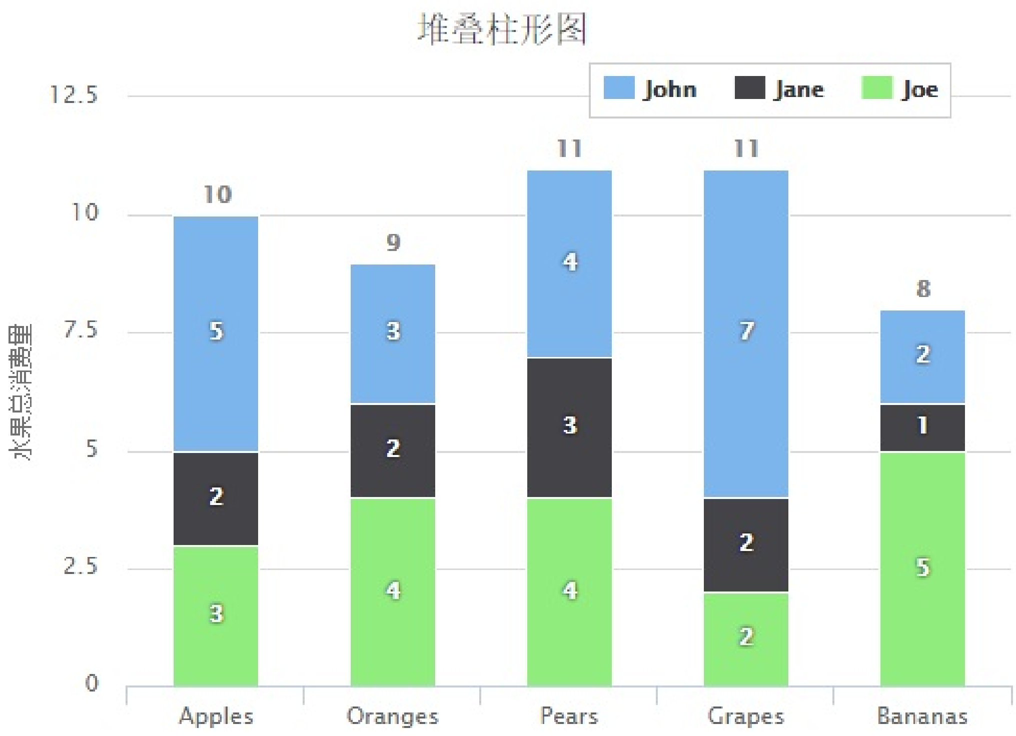
```
var plotOptions =  { column:  { stacking:  'normal', dataLabels:  
```

实例

文件名：highcharts_column_stacked.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



堆叠柱形图
John
Jane
Joe
12.5
10
7.5
5
2.5
0
水果总消费量
10
9
11
11
8
5
3
4
7
2
2
2
3
2
1
3
4
4
2
5
Apples
Oranges
Pears
Grapes
Bananas

Highcharts 堆叠组柱形图

以下实例演示了堆叠组柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 plotOptions 中添加 stacking 属性：

配置

plotOptions：数据点选项

plotOptions用于设置图表中的数据点相关属性。plotOptions根据各种图表类型，其属性设置略微有些差异。

配置图表堆叠设置 plotOptions.area.stacking 为 "percent"。如果禁用堆叠使用 null。

```
var plotOptions = { column: { stacking: 'normal', dataLabels: 
```

series 数据列项配置

配置堆叠组中每个对应的数据列项。

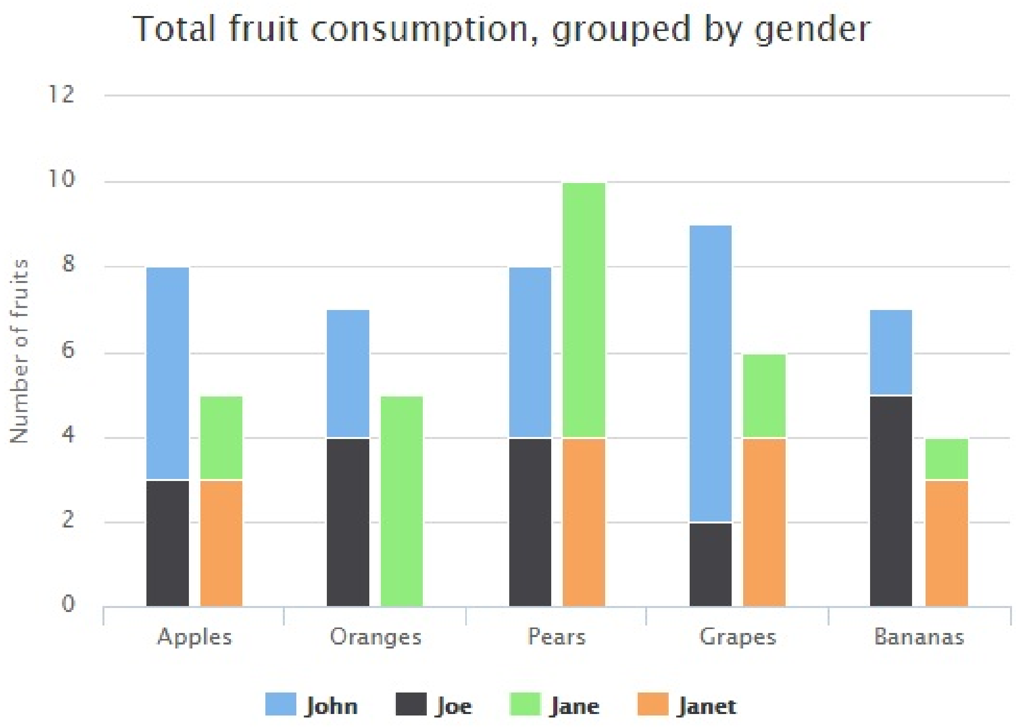
```
series: [{ name: 'John', data: [5, 3, 4, 7, 2], stack: 'mal
```

实例

文件名：highcharts_column_rotated.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title
```

以上实例输出结果为：



Total fruit consumption, grouped by gender
Number of fruits
0
2
4
6
8
10
12
Apples
Oranges
Pears
Grapes
Bananas
John
Joe
Jane
Janet

Highcharts 使用百分比的堆叠柱形图

以下实例演示了使用百分比的堆叠柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 plotOptions 中添加 stacking 属性：

配置

plotOptions：数据点选项

plotOptions用于设置图表中的数据点相关属性。plotOptions根据各种图表类型，其属性设置略微有些差异。

配置图表堆叠设置 plotOptions.area.stacking 为 "percent"。如果禁用堆叠使用null。

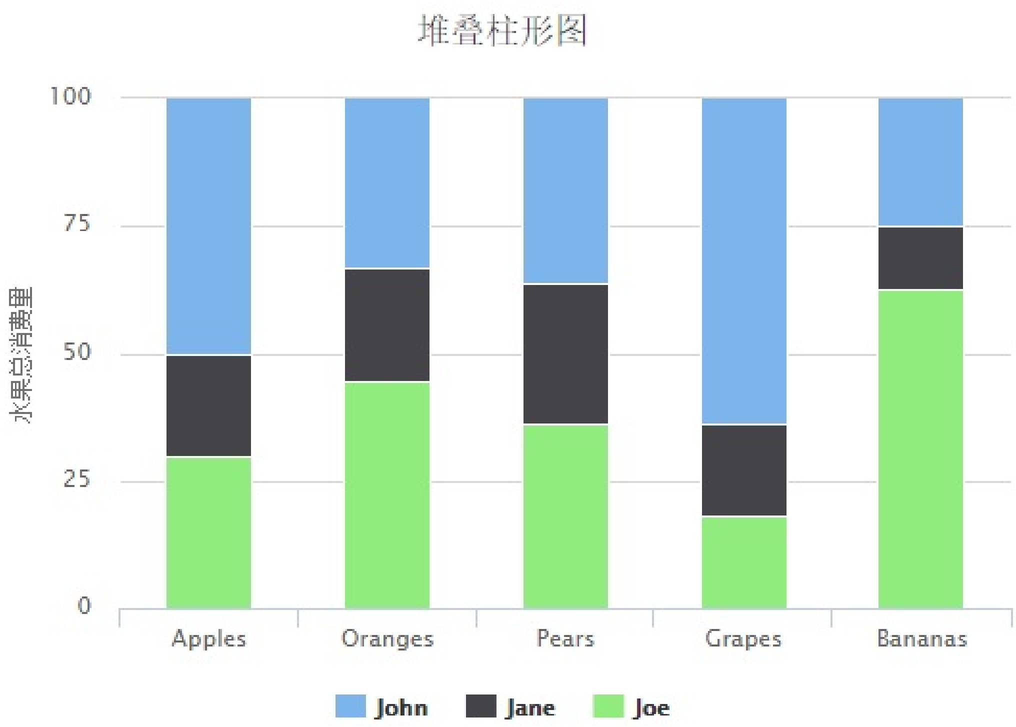
```
var plotOptions = { column: { stacking: 'percent' } };
```

实例

文件名：highcharts_column_percentage.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title
```

以上实例输出结果为：


堆叠柱形图
100
75
50
25
0
水果总消费量
Apples
Oranges
Pears
Grapes
Bananas
John
Jane
Joe

Highcharts 标签旋转柱形图

以下实例演示了标签旋转柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 dataLabels 中添加 rotation 属性：

配置

dataLabels：数据标签

dataLabels 定义图上是否显示数据标签。

文本旋转程度由 rotation 属性决定。我们还可以通过其他属性来定义文本标签的背景，间隔，边框等。

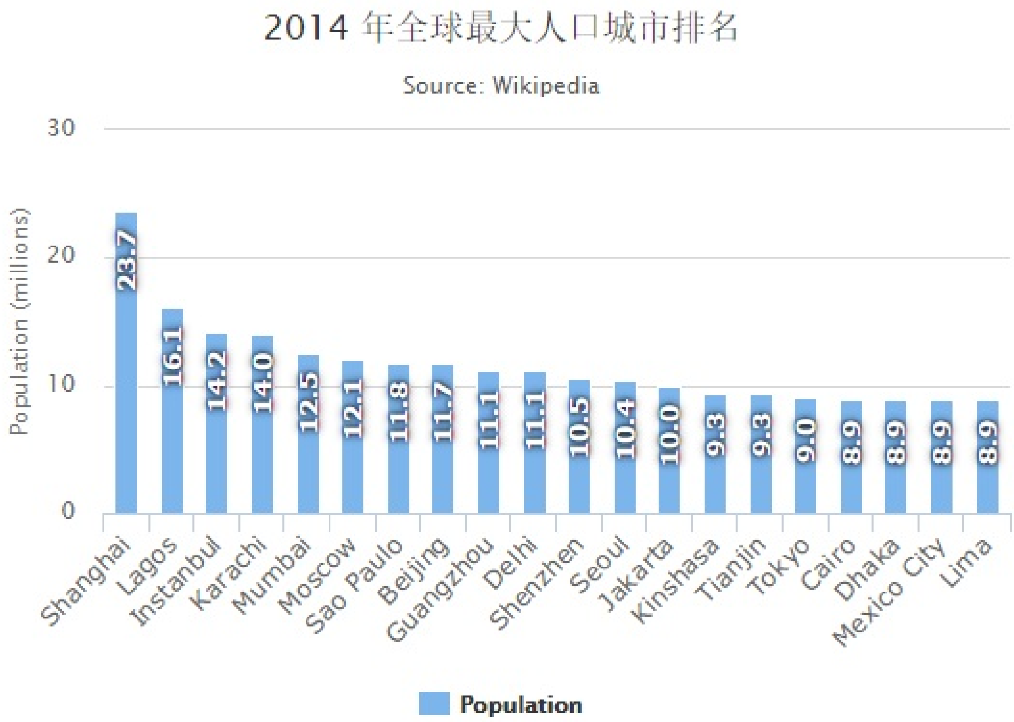
```
dataLabels =  { enabled:  true, rotation:  -90, color:  '#FFFFFF',
```

实例

文件名：highcharts_column_rotated.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


2014 年全球最大人口城市排名
Source: Wikipedia
Population (millions)
30
20
10
0
23.7
16.1
14.2
14.0
12.5
12.1
11.8
11.7
11.1
11.1
10.5
10.4
10.0
9.3
9.3
9.0
8.9
8.9
8.9
8.9
Shanghai
Lagos
Instanbul
Karachi
Mumbai
Moscow
Sao Paulo
Beijing
Guangzhou
Delhi
Shenzhen
Seoul
Jakarta
Kinshasa
Tianjin
Tokyo
Cairo
Dhaka
Mexico City
Lima
Population

Highcharts 堆叠组柱形图

以下实例演示了堆叠组柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 drilldown 中添加 series 属性：

配置

drilldown：向下钻取

drilldown 用于向下钻取数据，深入到其中的具体数据。

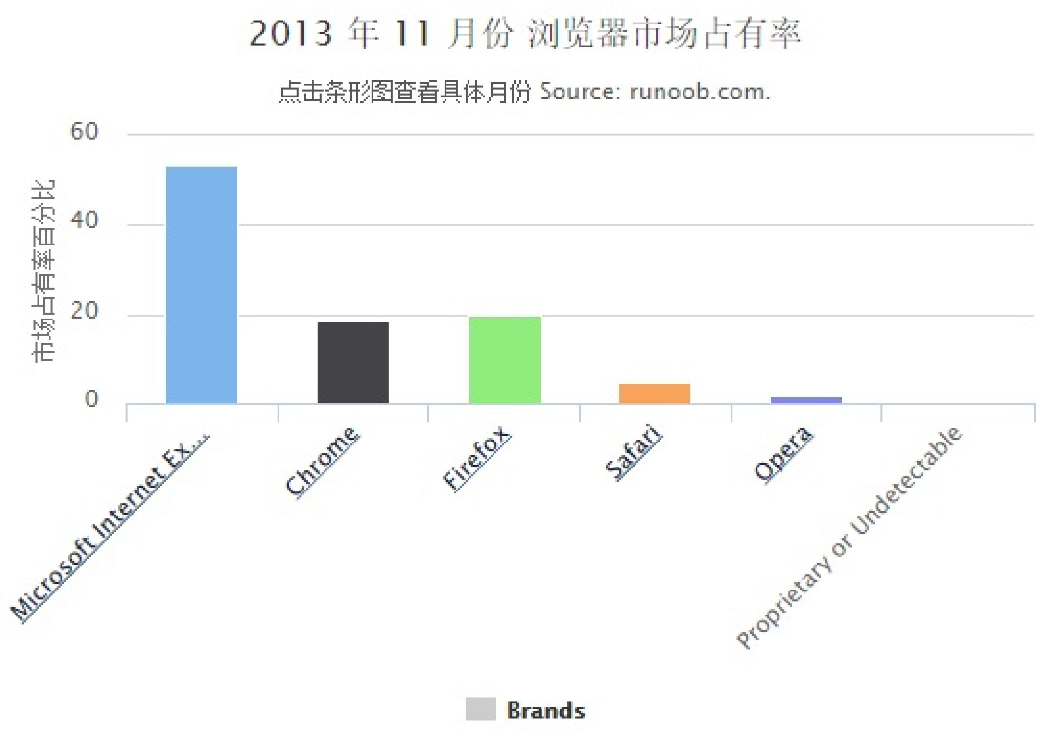
```
drilldown:  { series: drilldownSeries }
```

实例

文件名：highcharts_column_drilldown.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
Browser Version   Total Market Share
Microsoft Internet Explorer 8.0  26.61%
Microsoft Internet Explorer 9.0  16.96%
Chrome 18.0 8.01%
Chrome 19.0 7.73%
Firefox 12  6.72%
Microsoft Internet Explorer 6.0  6.40%
Firefox 11  4.72%
Microsoft Internet Explorer 7.0  3.55%
Safari 5.1  3.53%
Firefox 13  2.16%
Firefox 3.6 1.87%
Opera 11.x  1.30%
Chrome 17.0 1.13%
Firefox 10  0.90%
Safari 5.0  0.85%
Firefox 9.0 0.65%
Firefox 8.0 0.55%
Firefox 4.0 0.50%
Chrome 16.0 0.45%
Firefox 3.0 0.36%
Firefox 3.5 0.36%
Firefox 6.0 0.32%
Firefox 5.0 0.31%
Firefox 7.0 0.29%
Proprietary or Undetectable   0.29%
Chrome 18.0 - Maxthon Edition 0.26%
Chrome 14.0 0.25%
Chrome 20.0 0.24%
Chrome 15.0 0.18%
Chrome 12.0 0.16%
Opera 12.x  0.15%
Safari 4.0  0.14%
Chrome 13.0 0.13%
Safari 4.1  0.12%
Chrome 11.0 0.10%
Firefox 14  0.10%
Firefox 2.0 0.09%
Chrome 10.0 0.09%
Opera 10.x  0.09%
Microsoft Internet Explorer 8.0 - Tencent Traveler Edition  0.09%
< /pre><!-- 去掉 pre 前面的空格 -->  </body>  </html>
```

以上实例输出结果为：


2013 年 11 月份 浏览器市场占有率
点击条形图查看具体月份 Source: runoob.com.
市场占有率百分比
60
40
20
0
Microsoft Internet Ex...
Chrome
Firefox
Safari
Opera
Proprietary or Undetectable
Brands

Highcharts 固定布局柱形图

以下实例演示了固定布局柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 series 下添加 pointPlacement 和 pointPadding 配置。

series.pointPadding

控制每列之间的距离值，当highcharts图表宽度固定的情况下，此值越大，柱子宽度越小，反之相反。默认此值为0.1

series.pointPlacement

在柱形图中，当 pointPlacement 设置为 "on" 时, X轴上的点没有间隔。如果 pointPlacement 设置为 "between", 列之间按刻度进行布局。

在 Highcharts 3.0.2 后的版本, pointPlacement 可以支持数字，0 为轴上的值， -0.5 为当前值与前面一个值的间隔， 0.5 为 当前值与下一个值的间隔。 不同与文字设置选项，数字设置不影响轴之间的间隔。

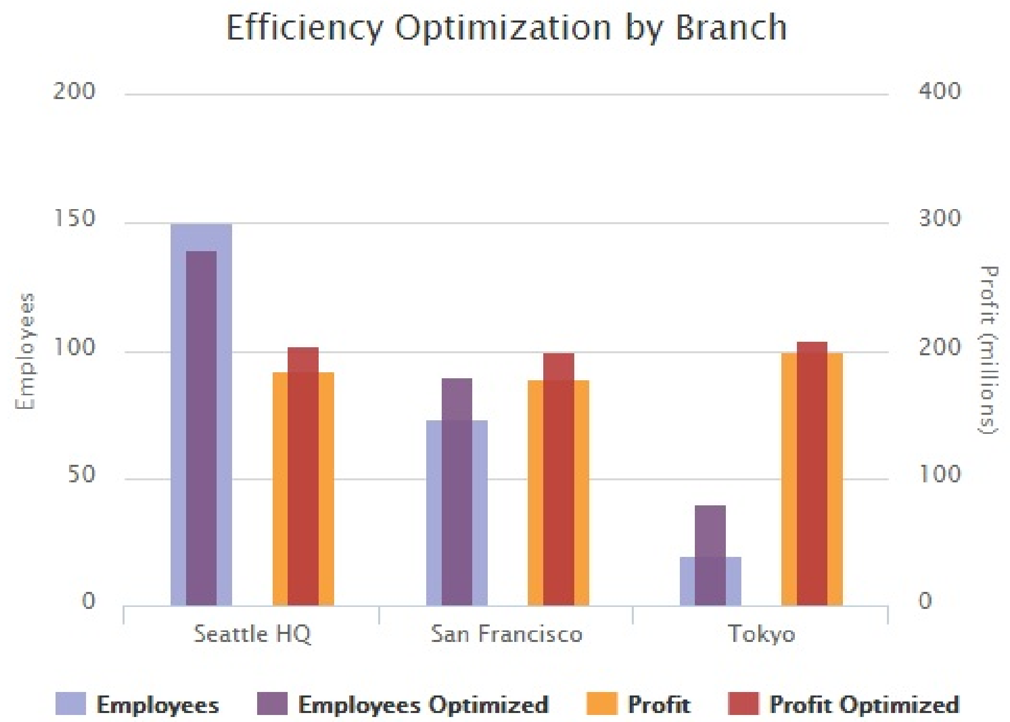
```
series:  { name:  'Employees', color:  'rgba(165,170,217,1)', data
```

实例

文件名：highcharts_column_fixed.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Efficiency Optimization by Branch
200
150
100
50
0
400
300
200
100
0
Employees
Profit (millions)
Seattle HQ
San Francisco
Tokyo
Employees
Employees Optimized
Profit
Profit Optimized

Highcharts 使用 HTML 表格数据的柱形图

以下实例演示了使用 HTML 表格数据的柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。在 data 下添加 table 配置。

data

数据模块提供了一些简单的接口用于添加数据，我们可以使用例如 CVS, HTML 表格或者网格视图中的数据。

data.table

Html 表格中设置id，并对应参数 data.table 即能解析数据。相关设置选项有 startRow, endRow, startColumn 和 endColumn 。

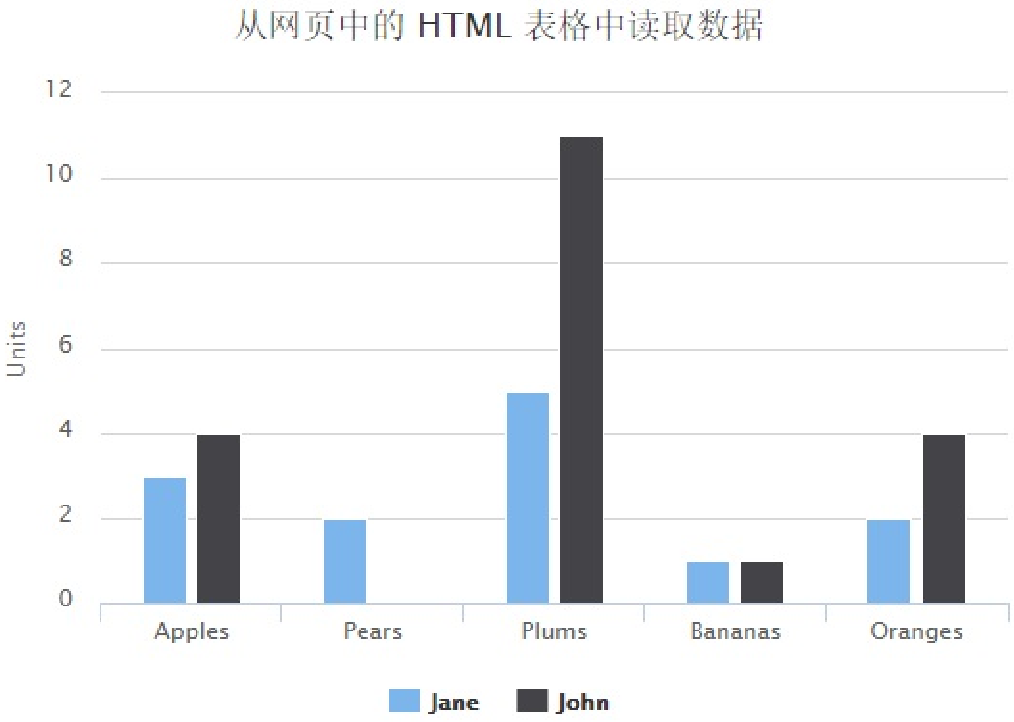
```
data:  { table:  'dataTable'  }
```

实例

文件名：highcharts_column_table.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title
```

以上实例输出结果为：


从网页中的 HTML 表格中读取数据
Units
12
10
8
6
4
2
0
Apples
Pears
Plums
Bananas
Oranges
Jane
John

Highcharts 区间柱形图

以下实例演示了区间柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

chart 配置

配置 chart 的 type 为 'columnrange' 。chart.type 描述了图表类型。默认值为 "line"。

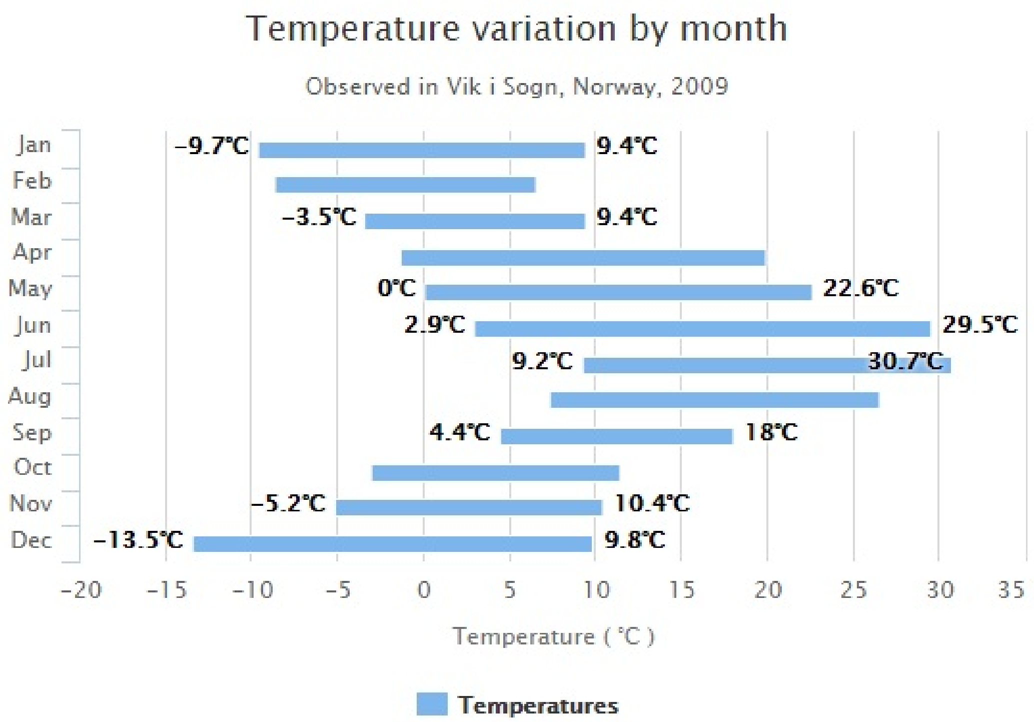
```
var chart =  { type:  'columnrange', inverted:  true  };
```

实例

文件名：highcharts_column_range.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 饼图

Highcharts 基本饼图

以下实例演示了基本饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

series 配置

设置 series 的 type 属性为 pie， series.type 描述了数据列类型。默认值为 "line"。

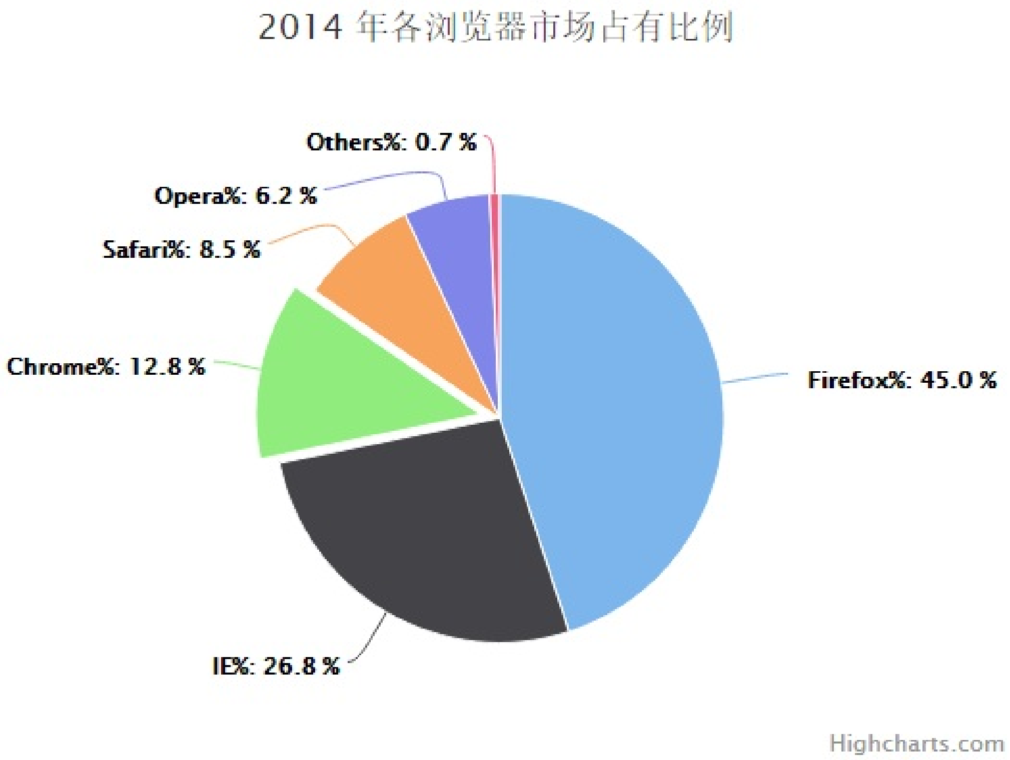
```
var series =  { type:  'pie'  };
```

实例

文件名：highcharts_pie_basic.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


2014 年各浏览器市场占有比例
Others%: 0.7 %
Opera%: 6.2 %
Safari%: 8.5 %
Chrome%: 12.8 %
Firefox%: 45.0 %
IE%: 26.8 %
Highcharts.com

Highcharts 显示图例饼图

以下实例演示了显示图例饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

series 配置

设置 series 的 type 属性为 pie ，series.type 描述了数据列类型。默认值为 "line"。

```
var series =  { type:  'pie'  };
```

plotOptions

plotOptions 使用 plotOptions.pie.showInLegend 属性来设置是否显示图例。

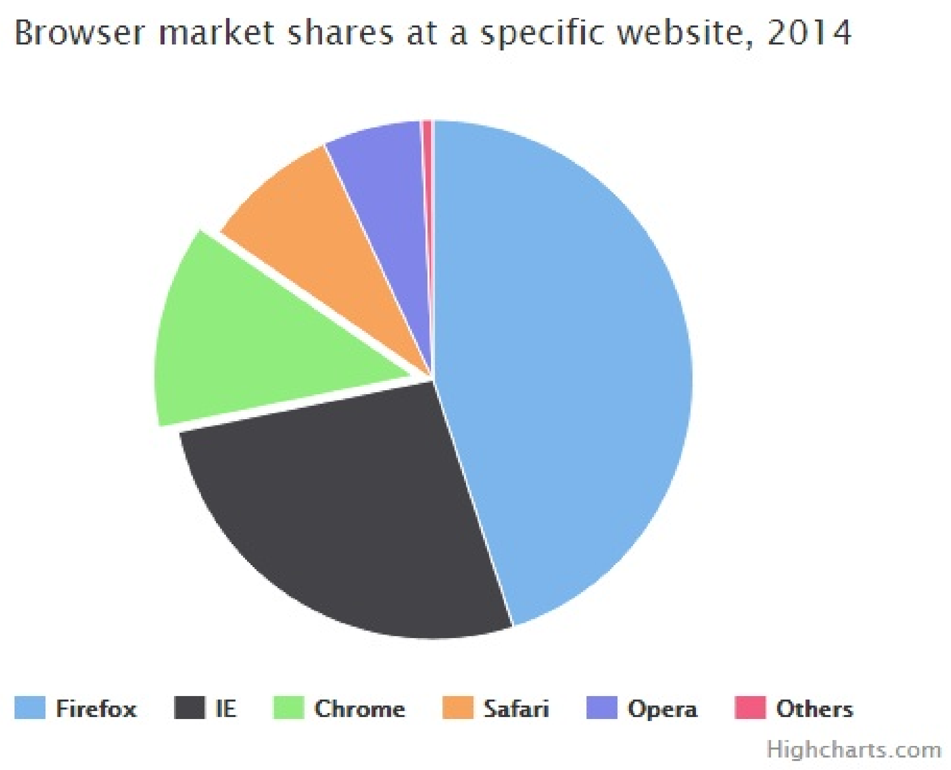
```
var plotOptions =  { pie:  { allowPointSelect:  true, cursor:  'poi
```

实例

文件名：highcharts_pie_legends.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Browser market shares at a specific website, 2014
Firefox
IE
Chrome
Safari
Opera
Others
Highcharts.com

Highcharts 圆环图

以下实例演示了圆环图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart 配置

设置 chart 的 type 属性为 pie chart.type 描述了图表类型。默认值为 "line"。

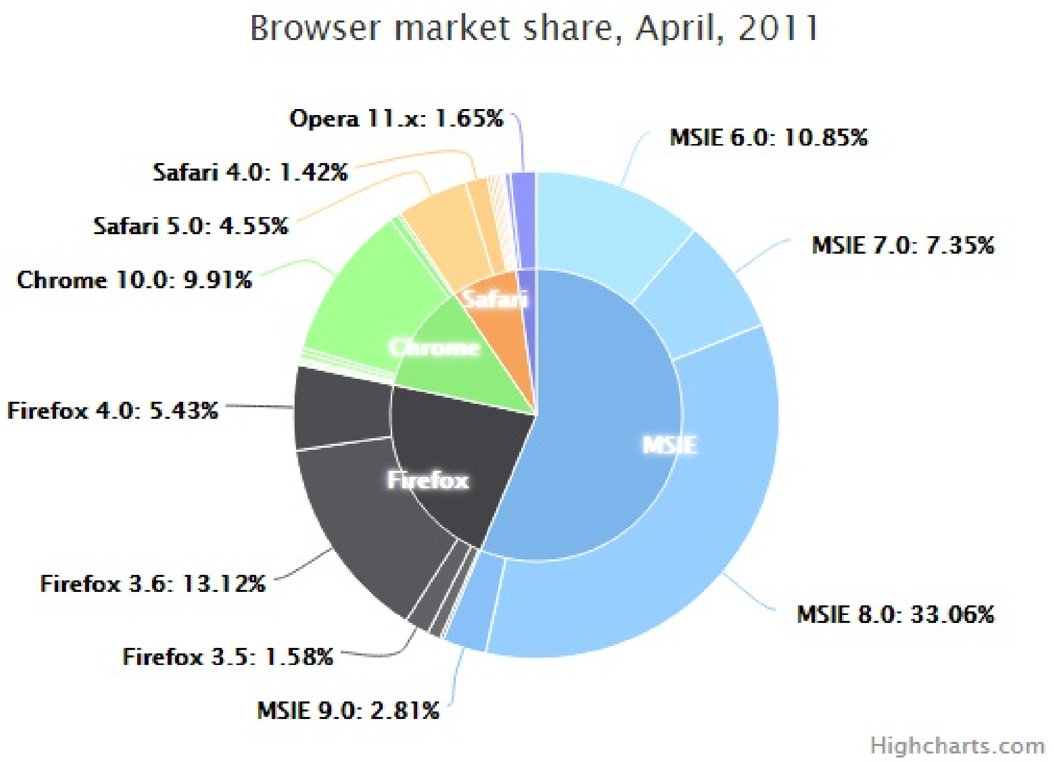
```
var chart = { type: 'pie' };
```

实例

文件名：highcharts_pie_donut.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Browser market share, April, 2011
Opera 11.x: 1.65%
MSIE 6.0: 10.85%
Safari 4.0: 1.42%
Safari 5.0: 4.55%
MSIE 7.0: 7.35%
Chrome 10.0: 9.91%
Safari
Chrome
Firefox 4.0: 5.43%
MSIE
Firefox
Firefox 3.6: 13.12%
MSIE 8.0: 33.06%
Firefox 3.5: 1.58%
MSIE 9.0: 2.81%
Highcharts.com

Highcharts 半圈圆环图

以下实例演示了半圈圆环图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

series 配置

设置 series 的 type 属性为 pie series.type 描述了数据列类型。默认值为 "line"。配置饼图大小使用 innerSize 属性并设置为innerSize: '50%'。

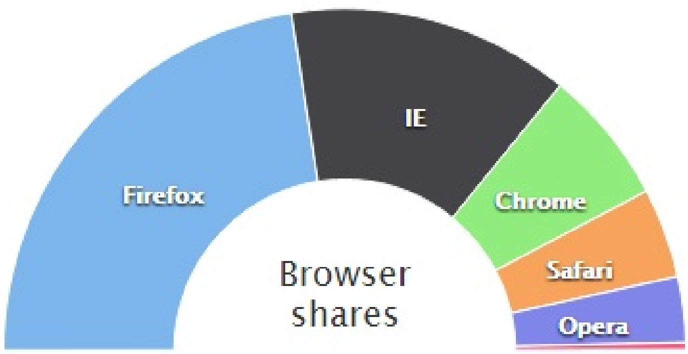
```
var series =  { type:  'pie', innerSize:  '50%'  };
```

实例

文件名：highcharts_pie_semicircle_donut.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



Highcharts 向下钻取饼图

以下实例演示了向下钻取饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

drilldown 配置

drilldown 用于向下钻取数据，通过点击某项深入到其中的具体数据。

```
drilldown:  { series: drilldownSeries }
```

实例

文件名：highcharts_pie_drilldown.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
Microsoft Internet Explorer 8.0    26.61%
Microsoft Internet Explorer 9.0    16.96%
Chrome 18.0    8.01%
Chrome 19.0    7.73%
Firefox 12    6.72%
Microsoft Internet Explorer 6.0    6.40%
Firefox 11    4.72%
Microsoft Internet Explorer 7.0    3.55%
Safari 5.1    3.53%
Firefox 13    2.16%
Firefox 3.6    1.87%
Opera 11.x    1.30%
Chrome 17.0    1.13%
Firefox 10    0.90%
Safari 5.0    0.85%
Firefox 9.0    0.65%
Firefox 8.0    0.55%
Firefox 4.0    0.50%
Chrome 16.0    0.45%
Firefox 3.0    0.36%
Firefox 3.5    0.36%
Firefox 6.0    0.32%
Firefox 5.0    0.31%
Firefox 7.0    0.29%
Proprietary or Undetectable    0.29%
Chrome 18.0 - Maxthon Edition    0.26%
Chrome 14.0    0.25%
Chrome 20.0    0.24%
Chrome 15.0    0.18%
Chrome 12.0    0.16%
Opera 12.x    0.15%
Safari 4.0    0.14%
Chrome 13.0    0.13%
Safari 4.1    0.12%
Chrome 11.0    0.10%
Firefox 14    0.10%
Firefox 2.0    0.09%
Chrome 10.0    0.09%
Opera 10.x    0.09%
Microsoft Internet Explorer 8.0 - Tencent Traveler Edition    0.09%
```

Highcharts 渐变饼图

以下实例演示了渐变饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

colors 配置

使用 Highcharts.getOptions().colors 属性来配置颜色。

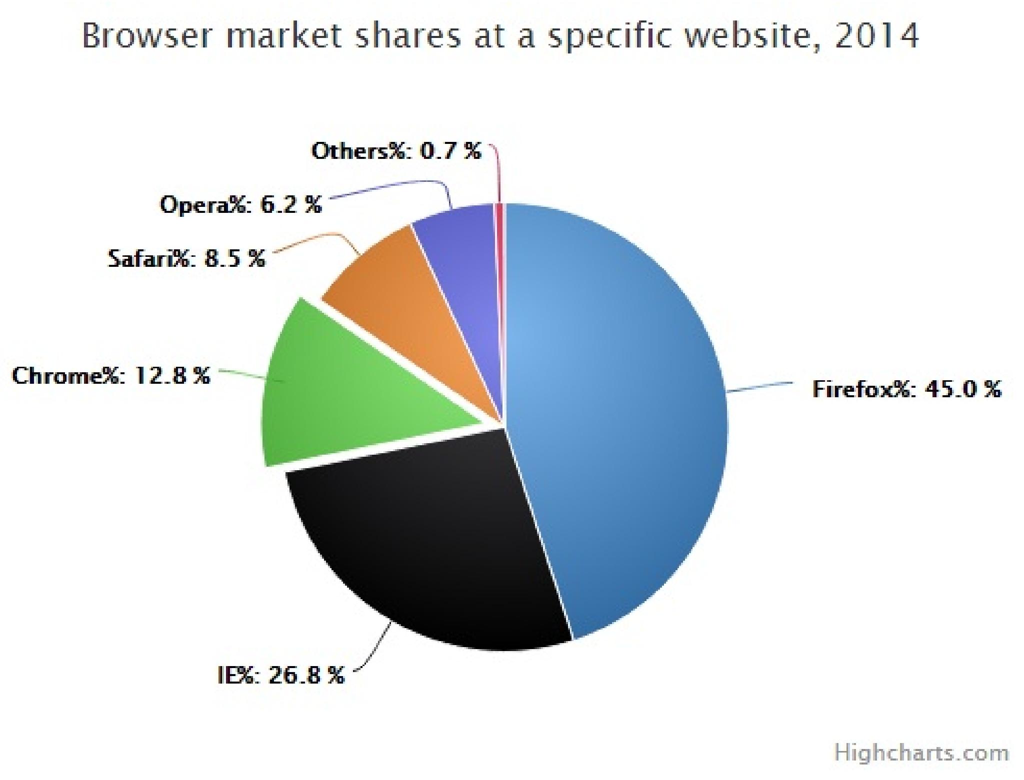
```
//颜色的填充  Highcharts.getOptions().colors =  Highcharts.map(Highc
```

实例

文件名：highcharts_pie_gradient.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Browser market shares at a specific website, 2014
Others%: 0.7 %
Opera%: 6.2 %
Safari%: 8.5 %
Chrome%: 12.8 %
Firefox%: 45.0 %
IE%: 26.8 %
Highcharts.com

Highcharts 单色饼图

以下实例演示了单色饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

colors 配置

使用 Highcharts.getOptions().plotOptions.pie.colors 属性来配置每个饼的颜色。

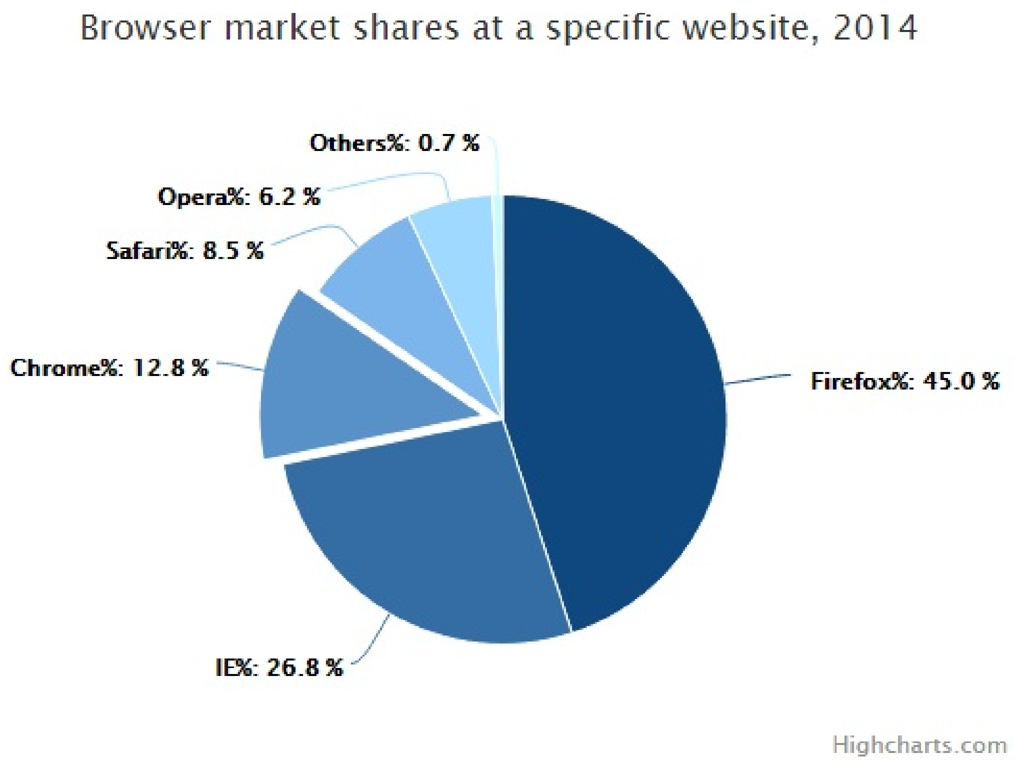
```
// 设置单色并为所有饼设置默认主题  Highcharts.getOptions().plotOptions.p
```

实例

文件名：highcharts_pie_monochrome.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Browser market shares at a specific website, 2014
Others%: 0.7 %
Opera%: 6.2 %
Safari%: 8.5 %
Chrome%: 12.8 %
Firefox%: 45.0 %
IE%: 26.8 %
Highcharts.com

Highcharts 散点图

本章节我们将为大家介绍 Highcharts 的散点图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下 Highcharts 的其他配置。

配置

chart 配置

配置 chart 的 type 为 'scatter' 。chart.type 描述了图表类型。默认值为 "line"。

chart.zoomType 属性可配置图表放大，通过拖动鼠标进行缩放，沿x轴或y轴进行缩放，可以设置为：'x','y','xy'。

```
var chart = {
   type: 'scatter',
   zoomType: 'xy'
};
```

实例

文件名：highcharts_scatter_basic.htm

```
<html>
<head>
<title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
   <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min
   <script src="/try/demo_source/highcharts.js"></script>
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {
   var chart = {
      type: 'scatter',
      zoomType: 'xy'
   };
   var title = {
      text: 'Height Versus Weight of 507 Individuals by Gender'
   };
   var subtitle = {
      text: 'Source: Heinz  2003'
   };
```

```
    var xAxis = {
       title: {
       enabled: true,
          text: 'Height (cm)'
       },
       startOnTick: true,
       endOnTick: true,
       showLastLabel: true
    };
    var yAxis = {
       title: {
          text: 'Weight (kg)'
       }
    };
    var legend = {
       layout: 'vertical',
       align: 'left',
       verticalAlign: 'top',
       x: 100,
       y: 70,
       floating: true,
       backgroundColor: (Highcharts.theme && Highcharts.theme.legenc
       borderWidth: 1
    }
    var plotOptions = {
       scatter: {
          marker: {
             radius: 5,
             states: {
                hover: {
                   enabled: true,
                   lineColor: 'rgb(100,100,100)'
                }
             }
          },
          states: {
             hover: {
                marker: {
                   enabled: false
                }
             }
          },
          tooltip: {
             headerFormat: '<b>{series.name}</b><br>',
             pointFormat: '{point.x} cm, {point.y} kg'
          }
       }
    };
    var series= [{
             name: 'Female',
             color: 'rgba(223, 83, 83, .5)',
             data: [[161.2, 51.6], [167.5, 59.0], [159.5, 49.2], [15
                 [170.0, 59.0], [159.1, 47.6], [166.0, 69.8], [176.2
```

```
                [172.5, 55.2], [170.9, 54.2], [172.9, 62.5], [153.4
                [147.2, 49.8], [168.2, 49.2], [175.0, 73.2], [157.0
                [159.5, 50.6], [175.0, 82.5], [166.8, 57.2], [176.5
                [174.0, 54.5], [173.0, 59.8], [179.9, 67.3], [170.5
                [154.4, 46.2], [162.0, 55.0], [176.5, 83.0], [160.0
                [162.1, 53.6], [170.0, 73.2], [160.2, 52.1], [161.3
                [168.9, 62.3], [163.8, 58.5], [167.6, 54.5], [160.0
                [167.6, 58.3], [165.1, 56.2], [160.0, 50.2], [170.0
                [167.6, 61.0], [160.7, 69.1], [163.2, 55.9], [152.4
                [168.3, 54.8], [180.3, 60.7], [165.5, 60.0], [165.0
                [156.0, 52.7], [160.0, 74.3], [163.0, 62.0], [165.7
                [162.0, 54.7], [166.0, 53.2], [174.0, 75.7], [172.7
                [151.1, 48.7], [164.5, 52.3], [163.5, 50.0], [152.0
                [164.0, 55.7], [161.2, 54.8], [155.0, 45.9], [170.0
                [170.0, 69.4], [162.5, 58.2], [170.3, 64.8], [164.1
                [163.2, 59.8], [154.5, 49.0], [159.8, 50.0], [173.2
                [161.4, 63.4], [169.0, 58.2], [166.2, 58.6], [159.4
                [159.0, 48.6], [162.8, 57.8], [159.0, 55.6], [179.8
                [161.0, 53.6], [151.1, 73.2], [168.2, 53.4], [168.9
                [171.8, 56.2], [178.0, 70.6], [164.3, 59.8], [163.0
                [166.8, 56.6], [172.7, 105.2], [163.5, 51.8], [169.
                [159.5, 47.6], [167.6, 63.0], [161.2, 55.2], [160.0
                [162.2, 50.2], [161.3, 60.2], [149.5, 44.8], [157.5
                [172.7, 62.0], [155.0, 49.2], [156.5, 67.2], [164.0
                [162.8, 58.0], [167.0, 59.8], [160.0, 54.8], [160.0
                [158.2, 46.4], [156.0, 64.4], [160.0, 48.8], [167.1
                [167.6, 57.8], [156.0, 54.6], [162.1, 59.2], [173.4
                [170.5, 64.5], [159.2, 51.8], [157.5, 56.0], [161.3
                [160.0, 59.5], [168.9, 56.8], [165.1, 64.1], [162.6
                [166.4, 55.0], [160.0, 55.9], [152.4, 60.4], [170.2
                [170.2, 55.9], [158.8, 55.5], [172.7, 69.5], [167.6
                [167.6, 65.9], [156.2, 58.6], [175.2, 66.8], [172.1
                [160.0, 55.9], [165.1, 59.1], [182.9, 81.8], [166.4
                [177.8, 60.0], [165.1, 58.2], [175.3, 72.7], [154.9
                [172.7, 75.9], [168.9, 55.0], [161.3, 57.3], [167.6
                [175.3, 65.5], [157.5, 48.6], [163.8, 58.6], [167.6
                [165.1, 62.7], [168.9, 56.6], [162.6, 53.9], [164.5
                [168.9, 62.0], [175.3, 63.6], [159.4, 53.2], [160.0
                [162.6, 70.5], [167.6, 54.5], [162.6, 54.5], [160.7
                [157.5, 63.6], [162.6, 54.5], [152.4, 47.3], [170.2
                [172.7, 70.5], [165.1, 60.9], [170.2, 63.6], [170.2
                [161.3, 70.5], [167.6, 52.7], [167.6, 62.7], [165.1
                [152.4, 67.3], [168.9, 63.0], [170.2, 73.6], [175.2
                [160.0, 55.4], [165.1, 104.1], [174.0, 55.5], [170.
                [167.6, 64.5], [167.6, 72.3], [167.6, 61.4], [154.9
                [175.3, 63.6], [171.4, 53.4], [157.5, 54.5], [165.1
                [174.0, 73.6], [162.6, 61.4], [174.0, 55.5], [162.6
                [156.2, 60.0], [149.9, 46.8], [169.5, 57.3], [160.0
                [169.5, 67.3], [160.0, 75.5], [172.7, 68.2], [162.6
                [176.5, 71.8], [164.4, 55.5], [160.7, 48.6], [174.0

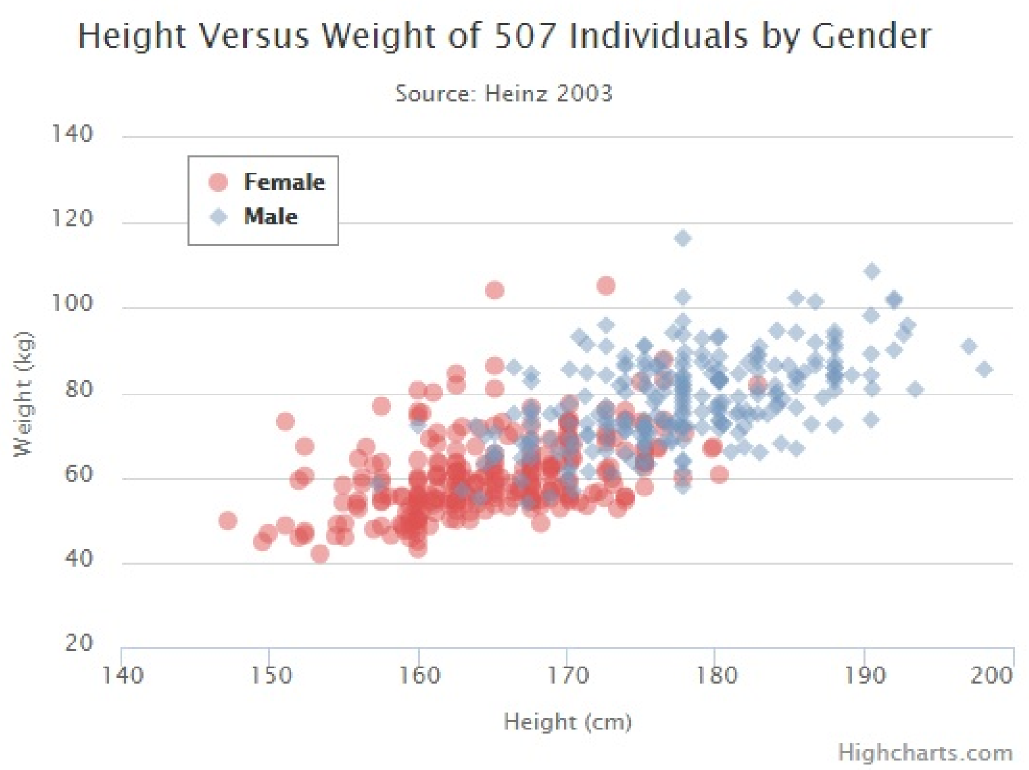
        }, {
            name: 'Male',
```

```
            color: 'rgba(119, 152, 191, .5)',
            data: [[174.0, 65.6], [175.3, 71.8], [193.5, 80.7], [18
                [181.5, 74.8], [184.0, 86.4], [184.5, 78.4], [175.0
                [180.0, 76.6], [177.8, 83.6], [192.0, 90.0], [176.0
                [184.0, 79.6], [192.7, 93.8], [171.5, 70.0], [173.0
                [176.0, 78.8], [180.5, 77.8], [172.7, 66.2], [176.0
                [178.0, 89.6], [180.3, 82.8], [180.3, 76.4], [164.5
                [183.5, 74.8], [175.5, 70.0], [188.0, 72.4], [189.2
                [170.0, 59.5], [182.0, 67.2], [170.0, 61.3], [177.8
                [186.7, 87.8], [171.4, 84.7], [172.7, 73.4], [175.3
                [182.9, 88.7], [188.0, 84.1], [177.2, 94.1], [172.1
                [169.5, 75.6], [174.0, 86.2], [172.7, 75.3], [182.2
                [163.0, 57.0], [171.5, 61.4], [184.2, 76.8], [174.0
                [177.0, 71.6], [186.0, 84.8], [167.0, 68.2], [171.8
                [167.0, 64.6], [177.8, 74.8], [164.5, 70.0], [192.0
                [171.2, 79.1], [181.6, 78.9], [167.4, 67.7], [181.1
                [174.5, 63.9], [177.5, 72.0], [170.5, 56.8], [182.4
                [180.1, 93.0], [175.5, 80.9], [180.6, 72.7], [184.4
                [180.6, 72.5], [177.0, 72.5], [177.1, 83.4], [181.6
                [175.0, 70.2], [174.0, 73.4], [165.1, 70.5], [177.0
                [176.5, 68.4], [169.4, 65.9], [182.1, 75.7], [179.8
                [184.9, 86.4], [177.3, 73.2], [167.4, 53.9], [178.1
                [157.2, 58.4], [180.3, 83.2], [170.2, 72.7], [177.8
                [165.1, 65.0], [186.7, 86.4], [165.1, 65.0], [174.0
                [185.4, 66.8], [177.8, 75.5], [180.3, 93.2], [180.3
                [177.8, 79.5], [177.8, 78.6], [177.8, 71.8], [177.8
                [188.0, 83.6], [198.1, 85.5], [175.3, 90.9], [166.4
                [166.4, 75.0], [177.8, 77.7], [179.7, 86.4], [172.7
                [185.4, 76.4], [168.9, 69.1], [167.6, 84.5], [175.3
                [190.5, 108.6], [177.8, 86.4], [190.5, 80.9], [177.
                [176.5, 80.2], [177.8, 72.0], [180.3, 71.4], [171.4
                [172.7, 76.8], [177.8, 63.6], [177.8, 80.9], [182.9
                [167.6, 68.6], [175.3, 67.7], [165.1, 66.4], [185.4
                [172.7, 95.9], [190.5, 84.1], [179.1, 87.3], [175.3
                [193.0, 95.9], [171.4, 91.4], [177.8, 81.8], [177.8
                [167.6, 82.7], [180.3, 75.5], [182.9, 79.5], [176.5
                [188.0, 84.1], [188.0, 85.9], [177.8, 81.8], [174.0
                [171.4, 70.0], [185.4, 81.8], [185.4, 84.1], [188.0
                [182.9, 89.1], [176.5, 85.0], [175.3, 69.1], [175.3
                [188.0, 82.7], [175.3, 86.4], [170.5, 67.7], [179.1
                [175.3, 70.9], [182.9, 75.0], [170.8, 93.2], [188.0
                [177.8, 61.4], [185.4, 94.1], [168.9, 75.0], [185.4
                [174.0, 73.9], [167.6, 66.8], [182.9, 87.3], [160.0
                [167.6, 75.5], [186.7, 101.4], [175.3, 91.1], [175.
                [175.3, 81.8], [179.1, 75.5], [181.6, 84.5], [177.8
                [177.8, 102.5], [184.2, 77.3], [179.1, 71.8], [176.
                [174.0, 70.9], [167.6, 64.5], [170.2, 77.3], [167.6
                [174.0, 80.0], [176.5, 82.3], [180.3, 73.6], [167.6
                [180.3, 73.2], [167.6, 76.3], [183.0, 65.9], [183.0
                [170.2, 62.3], [177.8, 82.7], [179.1, 79.1], [190.5
                [180.3, 83.2], [180.3, 83.2]]
        }
    ];
```

```
    var json = {};
    json.chart = chart;
    json.title = title;
    json.subtitle = subtitle;
    json.legend = legend;
    json.xAxis = xAxis;
    json.yAxis = yAxis;
    json.series = series;
    json.plotOptions = plotOptions;
    $('#container').highcharts(json);

 });
 </script>
 </body>
 </html>
```

以上实例输出结果为：



Highcharts 气泡图

本章节我们将为大家介绍 Highcharts 的气泡图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下 Highcharts 的其他配置。

配置

chart 配置

配置 chart 的 type 为 'bubble' 。chart.type 描述了图表类型。默认值为 "line"。

chart.zoomType 属性可配置图表放大 ，通过拖动鼠标进行缩放，沿x轴或y轴进行缩放，可以设置为：'x','y','xy'。

```
var chart = {
   type: 'bubble',
   zoomType: 'xy'
};
```

实例

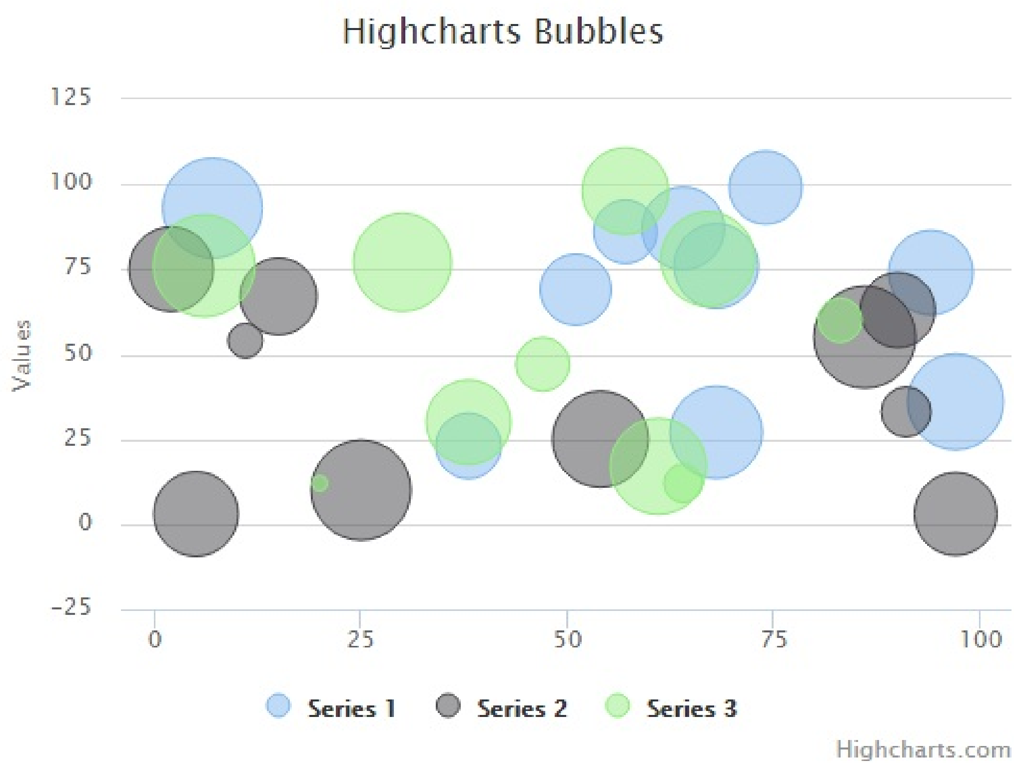
文件名：highcharts_bubble_basic.htm

```
<html>
<head>
<title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
   <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.
   <script src="/try/demo_source/highcharts.js"></script>
   <script src="http://code.highcharts.com/highcharts-more.js"></sc
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {
   var chart = {
      type: 'bubble',
      zoomType: 'xy'
   };
   var title = {
      text: 'Highcharts Bubbles'
   };
   var series= [{
            data: [[97, 36, 79], [94, 74, 60], [68, 76, 58], [64, 8
        }, {
            data: [[25, 10, 87], [2, 75, 59], [11, 54, 8], [86, 55,
        }, {
            data: [[47, 47, 21], [20, 12, 4], [6, 76, 91], [38, 30,
      }
   ];

   var json = {};
   json.chart = chart;
   json.title = title;
   json.series = series;
   $('#container').highcharts(json);

});
</script>
</body>
</html>
```

以上实例输出结果为：



3D 气泡图

series.marker

设置 series.marker 渐变让其看起来有 3D 的效果。

```
marker: {
   fillColor: {
      radialGradient: { cx: 0.4, cy: 0.3, r: 0.7 },
      stops: [
         [0, 'rgba(255,255,255,0.5)'],
         [1, Highcharts.Color(Highcharts.getOptions().colors[0]).se
      ]
   }
}
```

实例

文件名：highcharts_bubble_3d.htm

```
<html>
<head>
<title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
   <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min
```

```
    <script src="/try/demo_source/highcharts.js"></script>
    <script src="http://code.highcharts.com/highcharts-more.js"></sc
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {
   var chart = {
      type: 'bubble',
      plotBorderWidth: 1,
      zoomType: 'xy'
   };
   var title = {
      text: 'Highcharts bubbles with radial gradient fill'
   };
   var xAxis = {
      gridLineWidth: 1
   };

   var yAxis = {
      startOnTick: false,
      endOnTick: false
   };

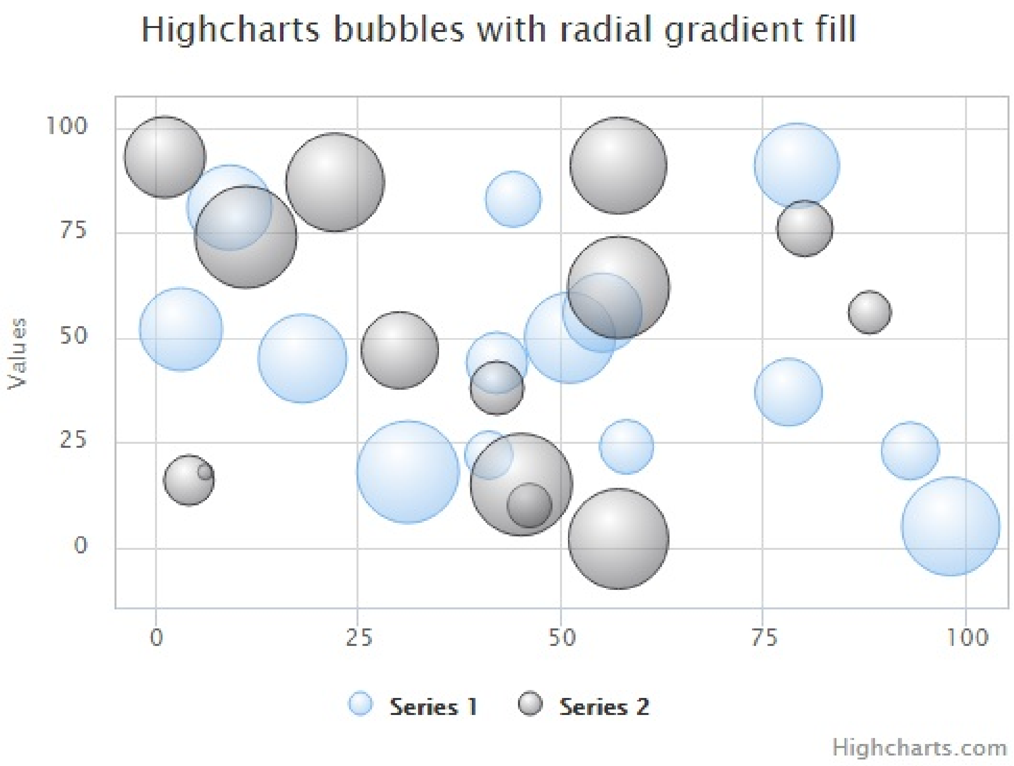
   var series= [{
           data: [
                [9, 81, 63],
                [98, 5, 89],
                [51, 50, 73],
                [41, 22, 14],
                [58, 24, 20],
                [78, 37, 34],
                [55, 56, 53],
                [18, 45, 70],
                [42, 44, 28],
                [3, 52, 59],
                [31, 18, 97],
                [79, 91, 63],
                [93, 23, 23],
                [44, 83, 22]
            ],
            marker: {
                fillColor: {
                    radialGradient: { cx: 0.4, cy: 0.3, r: 0.7 },
                    stops: [
                        [0, 'rgba(255,255,255,0.5)'],
                        [1, Highcharts.Color(Highcharts.getOptions(
                    ]
                }
            }
        }, {
            data: [
                [42, 38, 20],
```

```
                [6, 18, 1],
                [1, 93, 55],
                [57, 2, 90],
                [80, 76, 22],
                [11, 74, 96],
                [88, 56, 10],
                [30, 47, 49],
                [57, 62, 98],
                [4, 16, 16],
                [46, 10, 11],
                [22, 87, 89],
                [57, 91, 82],
                [45, 15, 98]
            ],
            marker: {
                fillColor: {
                    radialGradient: { cx: 0.4, cy: 0.3, r: 0.7 },
                    stops: [
                        [0, 'rgba(255,255,255,0.5)'],
                        [1, Highcharts.Color(Highcharts.getOptions(
                    ]
                }
            }
      }
   ];

   var json = {};
   json.chart = chart;
   json.title = title;
   json.xAxis = xAxis;
   json.yAxis = yAxis;
   json.series = series;
   $('#container').highcharts(json);

});
</script>
</body>
</html>
```

以上实例输出结果为：


Highcharts bubbles with radial gradient fill
Values
100
75
50
25
0
0
25
50
75
100
Series 1
Series 2
Highcharts.com

Highcharts 动态图

本章节我们将为大家介绍 Highcharts 的动态图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下 Highcharts 的其他配置。

每秒更新曲线图

chart.events

chart.event 属性中添加 load 方法（图表加载事件）。在 1000 毫秒内随机产生数据点并生成图表。

```
chart: {
   events: {
      load: function () {
         // 图表每秒更新一次
         var series = this.series[0];
         setInterval(function () {
            var x = (new Date()).getTime(), // 当期时间
            y = Math.random();
            series.addPoint([x, y], true, true);
         }, 1000);
      }
   }
}
```

实例

文件名：highcharts_dynamic_spline.htm

```
<html>
<head>
<title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
   <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min
   <script src="/try/demo_source/highcharts.js"></script>
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {
   var chart = {
      type: 'spline',
      animation: Highcharts.svg, // don't animate in IE < IE 10.
```

```
      marginRight: 10,
      events: {
         load: function () {
            // set up the updating of the chart each second
            var series = this.series[0];
            setInterval(function () {
               var x = (new Date()).getTime(), // current time
               y = Math.random();
               series.addPoint([x, y], true, true);
            }, 1000);
         }
      }
   };
   var title = {
      text: 'Live random data'
   };
   var xAxis = {
      type: 'datetime',
      tickPixelInterval: 150
   };
   var yAxis = {
      title: {
         text: 'Value'
      },
      plotLines: [{
         value: 0,
         width: 1,
         color: '#808080'
      }]
   };
   var tooltip = {
      formatter: function () {
      return '<b>' + this.series.name + '</b><br/>' +
         Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) + '<br/
         Highcharts.numberFormat(this.y, 2);
      }
   };
   var plotOptions = {
      area: {
         pointStart: 1940,
         marker: {
            enabled: false,
            symbol: 'circle',
            radius: 2,
            states: {
               hover: {
                 enabled: true
               }
            }
         }
      }
   };
   var legend = {
```

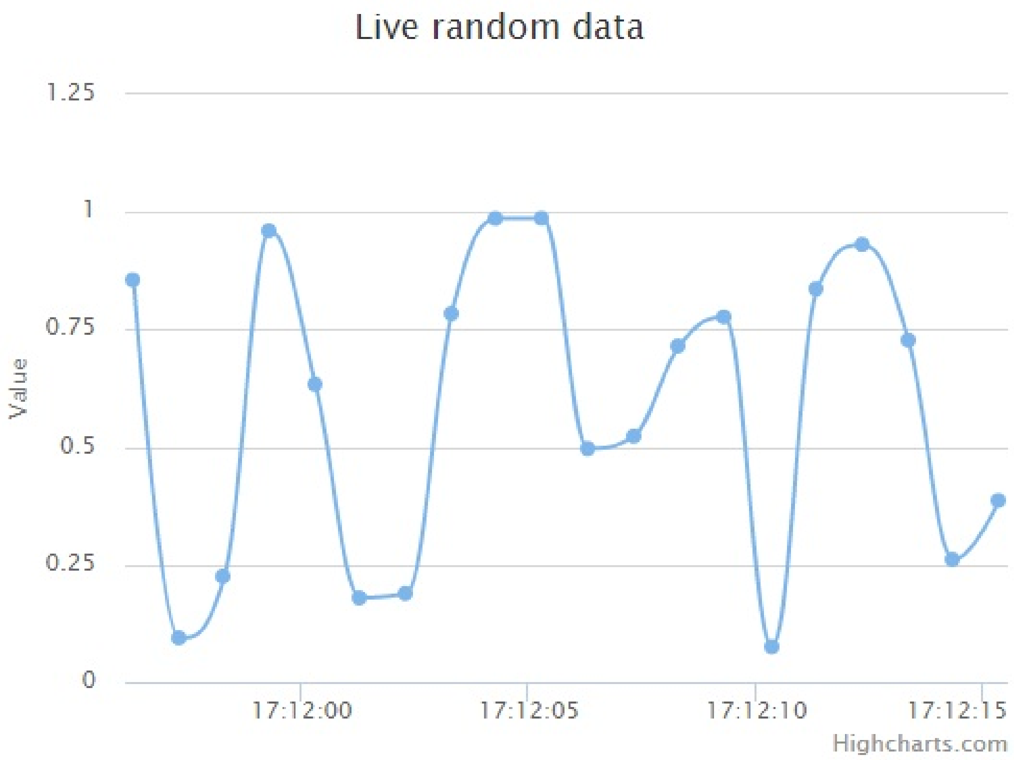
```
         enabled: false
      };
      var exporting = {
         enabled: false
      };
      var series= [{
         name: 'Random data',
         data: (function () {
            // generate an array of random data
            var data = [],time = (new Date()).getTime(),i;
            for (i = -19; i <= 0; i += 1) {
               data.push({
                  x: time + i * 1000,
                  y: Math.random()
               });
            }
            return data;
         }())
      }];

      var json = {};
      json.chart = chart;
      json.title = title;
      json.tooltip = tooltip;
      json.xAxis = xAxis;
      json.yAxis = yAxis;
      json.legend = legend;
      json.exporting = exporting;
      json.series = series;
      json.plotOptions = plotOptions;

      Highcharts.setOptions({
         global: {
            useUTC: false
         }
      });
      $('#container').highcharts(json);

   });
   </script>
   </body>
   </html>
```

以上实例输出结果为：



通过点击添加数据

chart.events

在 chart.event 属性中添加 click 方法（整个图表的绘图区上所发生的点击事件）。该方法在图表绘图区上发生点击时会添加新的数据点。

```
chart: {
    events: {
        click: function (e) {
            // 获取点击坐标和数据项
            var x = e.xAxis[0].value,
            y = e.yAxis[0].value,
            series = this.series[0];
            // 添加点击的坐标
            series.addPoint([x, y]);
        }
    }
}
```

实例

文件名：highcharts_dynamic_click.htm

```
<html>
```

```
<head>
<title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
   <script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.
   <script src="/try/demo_source/highcharts.js"></script>
</head>
<body>
<div id="container" style="width: 550px; height: 400px; margin: 0 a
<script language="JavaScript">
$(document).ready(function() {
   var chart = {
      type: 'scatter',
      margin: [70, 50, 60, 80],
      events: {
         click: function (e) {
            // find the clicked values and the series
            var x = e.xAxis[0].value,
            y = e.yAxis[0].value,
            series = this.series[0];
            // Add it
            series.addPoint([x, y]);
         }
      }
   };
   var title = {
      text: 'User supplied data'
   };
   var subtitle = {
      text: 'Click the plot area to add a point. Click a point to r
   };
   var xAxis = {
      gridLineWidth: 1,
      minPadding: 0.2,
      maxPadding: 0.2,
      maxZoom: 60
   };
   var yAxis = {
      title: {
         text: 'Value'
      },
      minPadding: 0.2,
      maxPadding: 0.2,
      maxZoom: 60,
      plotLines: [{
         value: 0,
         width: 1,
         color: '#808080'
      }]
   };
   var legend = {
      enabled: false
   };
   var exporting = {
      enabled: false
```

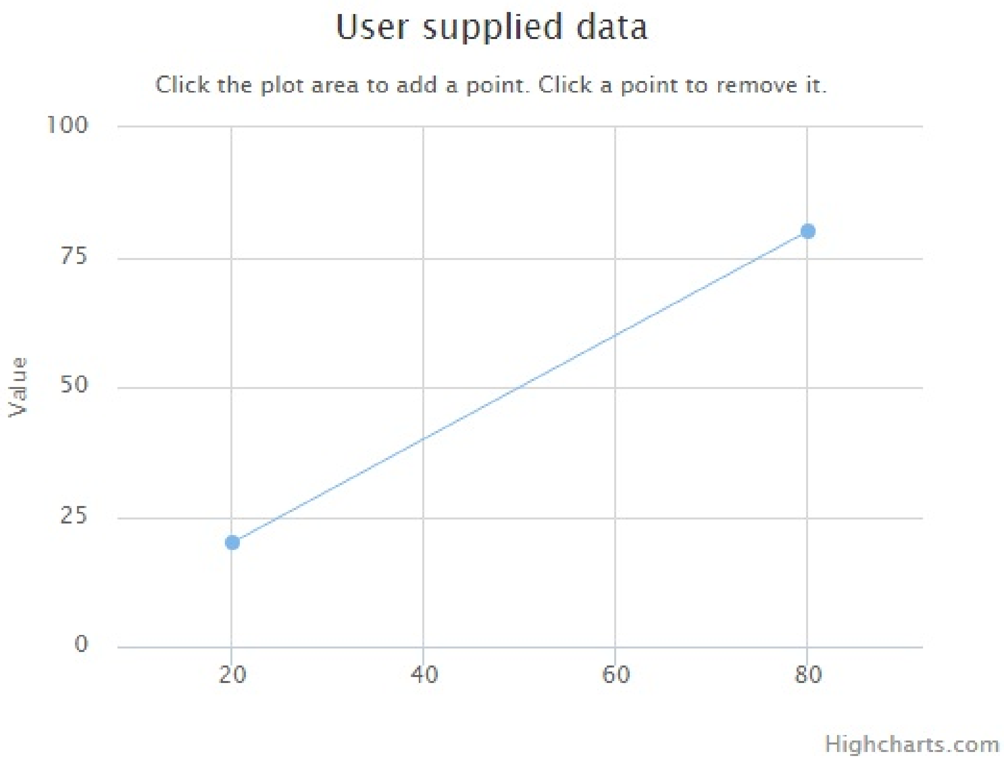
```
    };
    var plotOptions = {
       series: {
          lineWidth: 1,
          point: {
             events: {
                'click': function () {
                   if (this.series.data.length > 1) {
                      this.remove();
                   }
                }
             }
          }
       }
    };

    var series= [{
       data: [[20, 20], [80, 80]]
    }];

    var json = {};
    json.chart = chart;
    json.title = title;
    json.subtitle = subtitle;
    json.xAxis = xAxis;
    json.yAxis = yAxis;
    json.legend = legend;
    json.exporting = exporting;
    json.series = series;
    json.plotOptions = plotOptions;
    $('#container').highcharts(json);

});
</script>
</body>
</html>
```

以上实例输出结果为：


User supplied data
Click the plot area to add a point. Click a point to remove it.
Value
100
75
50
25
0
20
40
60
80
Highcharts.com

Highcharts 组合图

Highcharts 柱形图，线条图，饼图组合

以下实例演示了柱形图，线条图，饼图的组合。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

series 配置

设置 series 的 type 属性为 column/line/pie ， series.type 描述了数据列类型。默认值为 "line"。

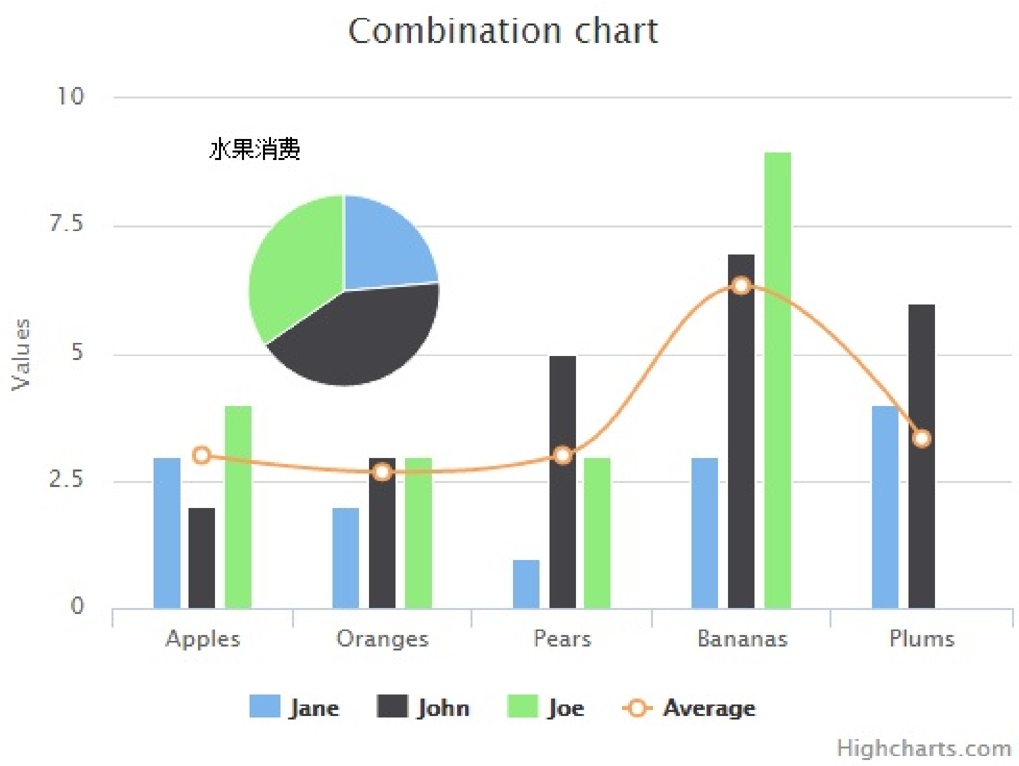
```
var series =  { type:  'column'  };
```

实例

文件名：highcharts_combinations_column.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


Combination chart
水果消费
Values
10
7.5
5
2.5
0
Apples
Oranges
Pears
Bananas
Plums
Jane
John
Joe
Average
Highcharts.com

Highcharts 双Y轴, 柱形图，线条图组合

以下实例演示了双Y轴, 柱形图，线条图组合。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

yAxis 配置

在 yAxis 属性中添加两条Y轴的值 。

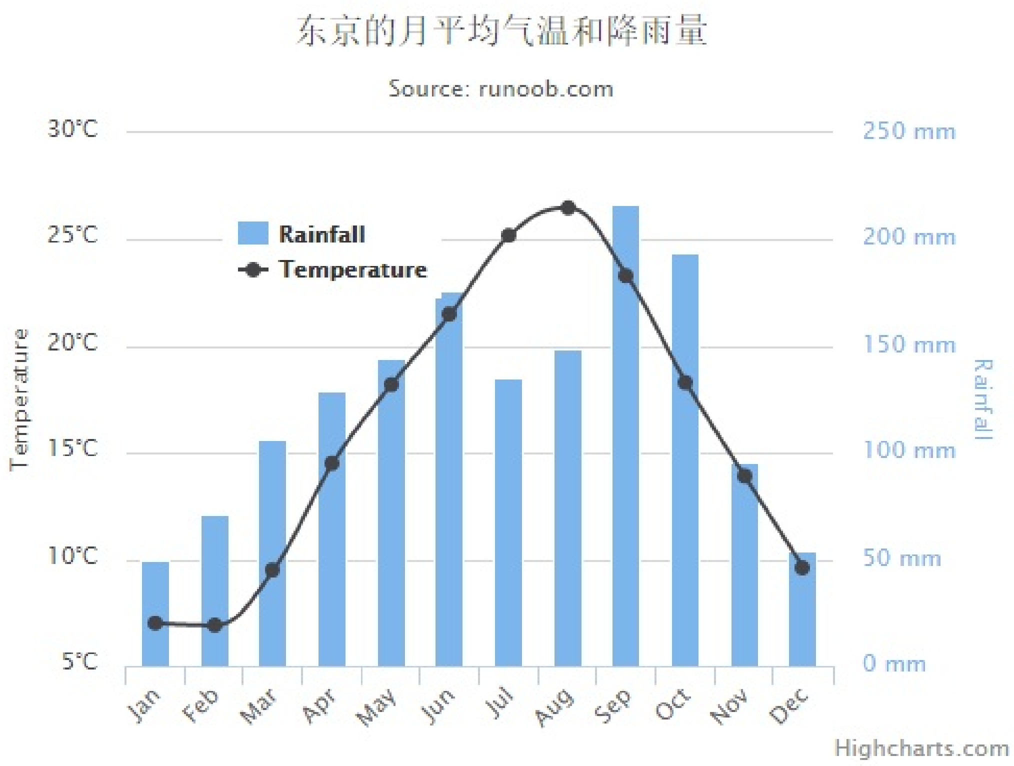
```
var yAxis =  [{  // 第一条Y轴  },  {  // 第二条Y轴  }  }]
```

实例

文件名：highcharts_combinations_dualaxes.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


东京的月平均气温和降雨量
Source: runoob.com
30°C
25°C
20°C
15°C
10°C
5°C
Temperature
250 mm
200 mm
150 mm
100 mm
50 mm
0 mm
Rainfall
Rainfall
Temperature
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Highcharts.com

Highcharts 多Y轴组合

以下实例演示了多Y轴的组合。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

yAxis 配置

在 yAxis 属性中添加多条Y轴的值 。

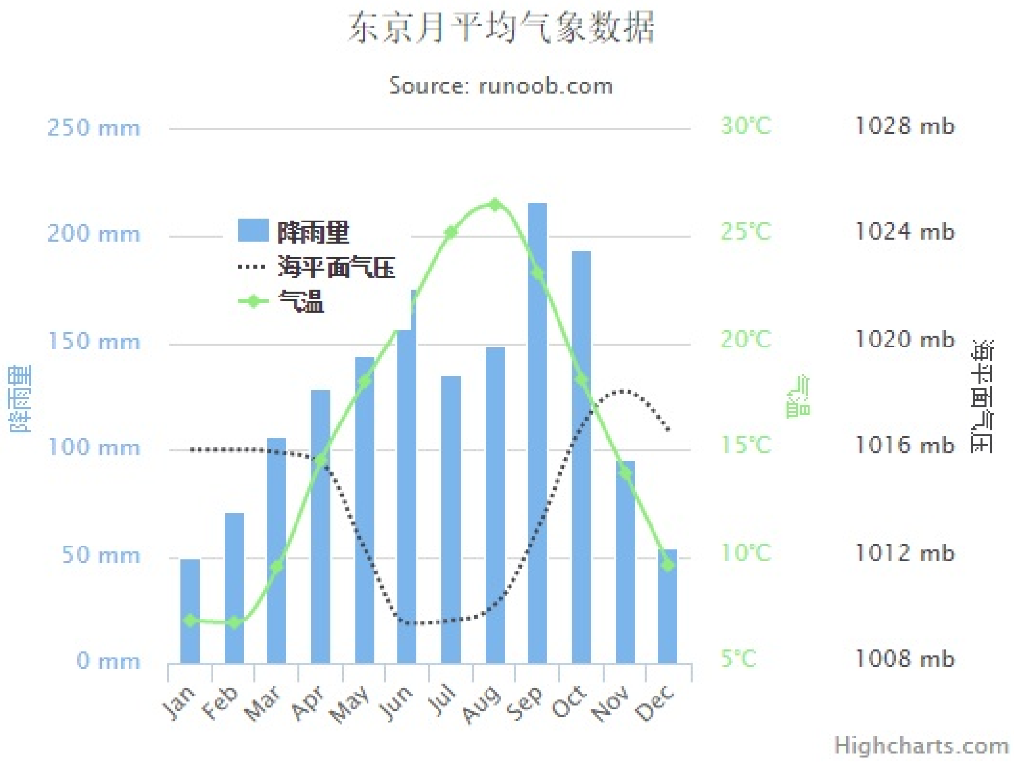
```
var yAxis =  [{  // 第一条Y轴  },  {  // 第二条Y轴  },{  // 第三题Y轴
```

实例

文件名：highcharts_combinations_axes.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


东京月平均气象数据
Source: runoob.com
250 mm
200 mm
150 mm
100 mm
50 mm
0 mm
降雨量
30°C
25°C
20°C
15°C
10°C
5°C
气温
1028 mb
1024 mb
1020 mb
1016 mb
1012 mb
1008 mb
海平面气压
降雨量
海平面气压
气温
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Highcharts.com

Highcharts 散点图上添加回归线

以下实例演示了散点图上添加回归线。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

series 配置

设置 series 的 type 属性为 line/scatter， series.type 描述了数据列类型。默认值为 "line"。

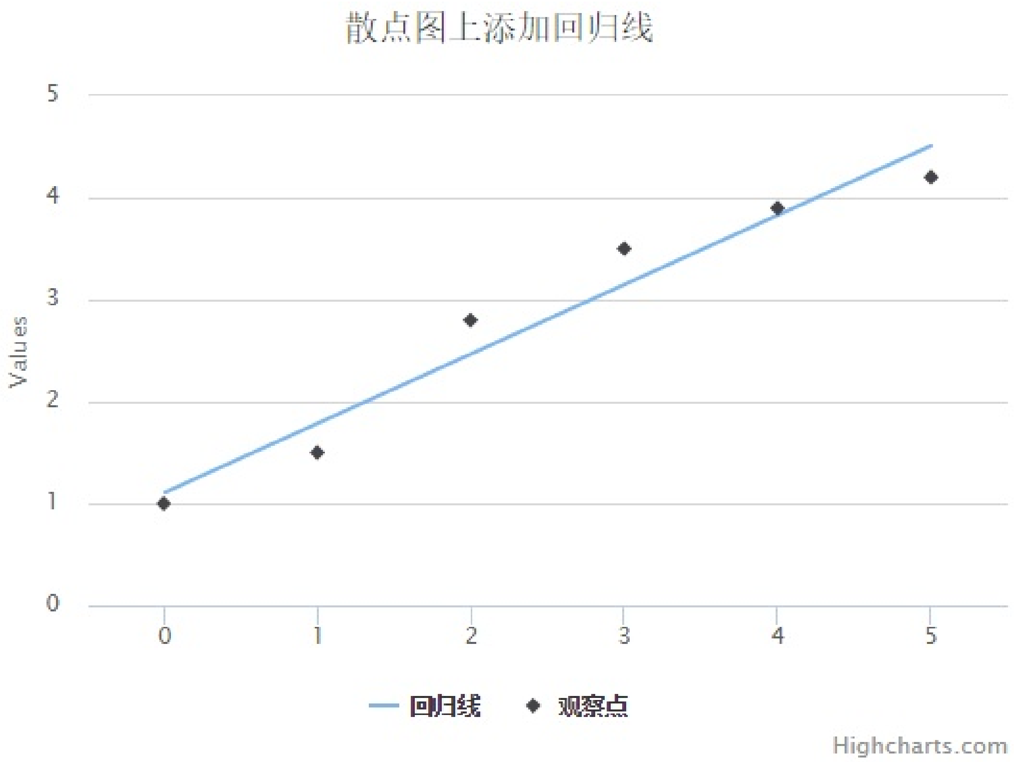
```
var series =  { type:  'scatter'  };
```

实例

文件名：highcharts_combinations_scatter.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


散点图上添加回归线
Values
5
4
3
2
1
0
0
1
2
3
4
5
回归线
观察点
Highcharts.com

Highcharts 3D图

Highcharts 3D柱形图

以下实例演示了3D柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.options3d 配置

以下列出了 3D 图的基本配置，设置 chart 的 type 属性为 column，options3d 选项可设置三维效果。

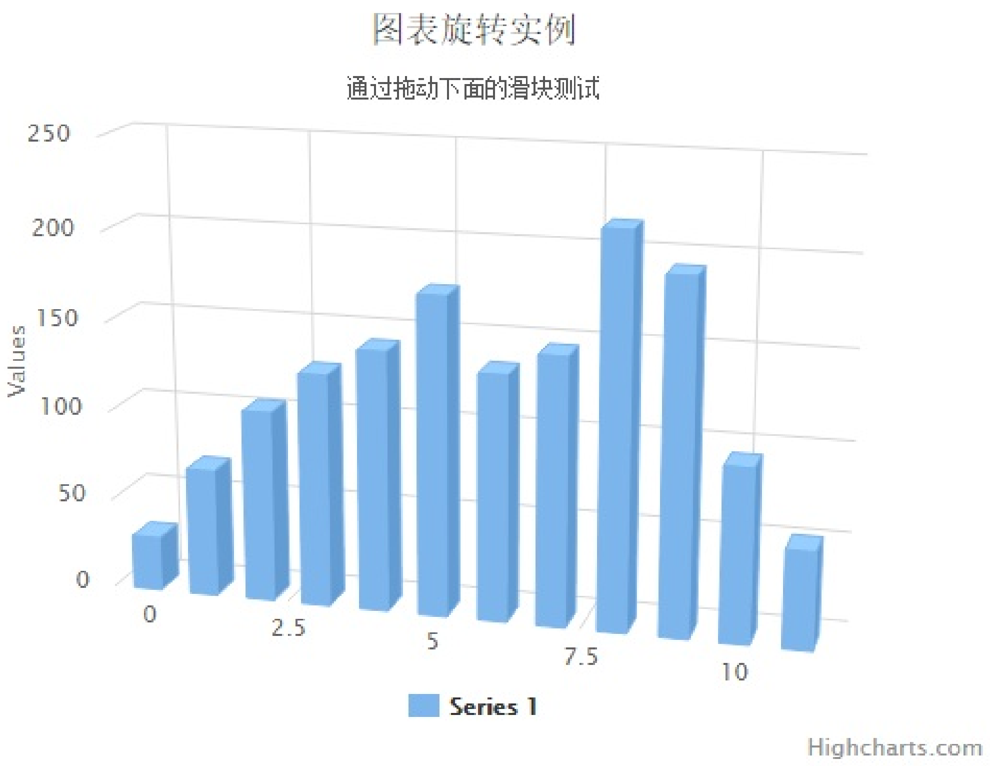
```
var chart =  { type:  'column', options3d:  { enabled:  true,  //显
```

实例

文件名：highcharts_3d_column.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



图表旋转实例
通过拖动下面的滑块测试
250
200
150
100
50
0
Values
0
2.5
5
7.5
10
Series 1
Highcharts.com

Highcharts 带空值(null)和0的3D柱形图

以下实例演示了带空值(null)和0的3D柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.options3d 配置

以下列出了 3D 图的基本配置，设置 chart 的 type 属性为 column，options3d 选项可设置三维效果。

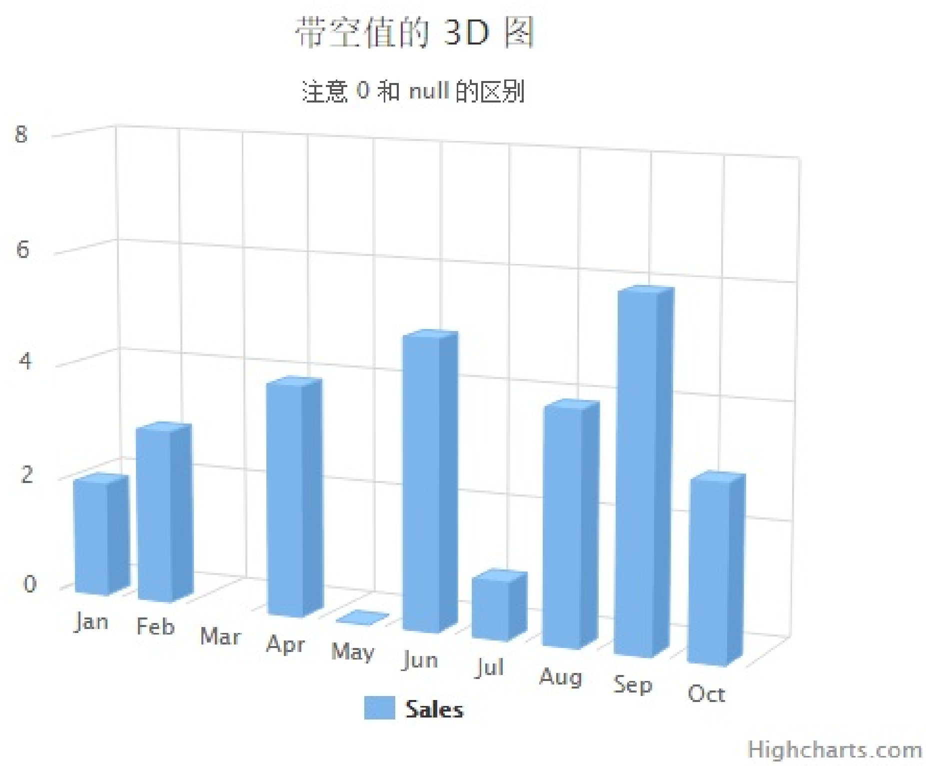
```
var chart = { type: 'column', options3d: { enabled: true, //显
```

实例

文件名：highcharts_3d_column_null.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：



带空值的 3D 图
注意 0 和 null 的区别
8
6
4
2
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Sales
Highcharts.com

Highcharts 堆叠3D柱形图

以下实例演示了堆叠3D柱形图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.options3d 配置

以下列出了 3D 图的基本配置，设置 chart 的 type 属性为 column，options3d 选项可设置三维效果。

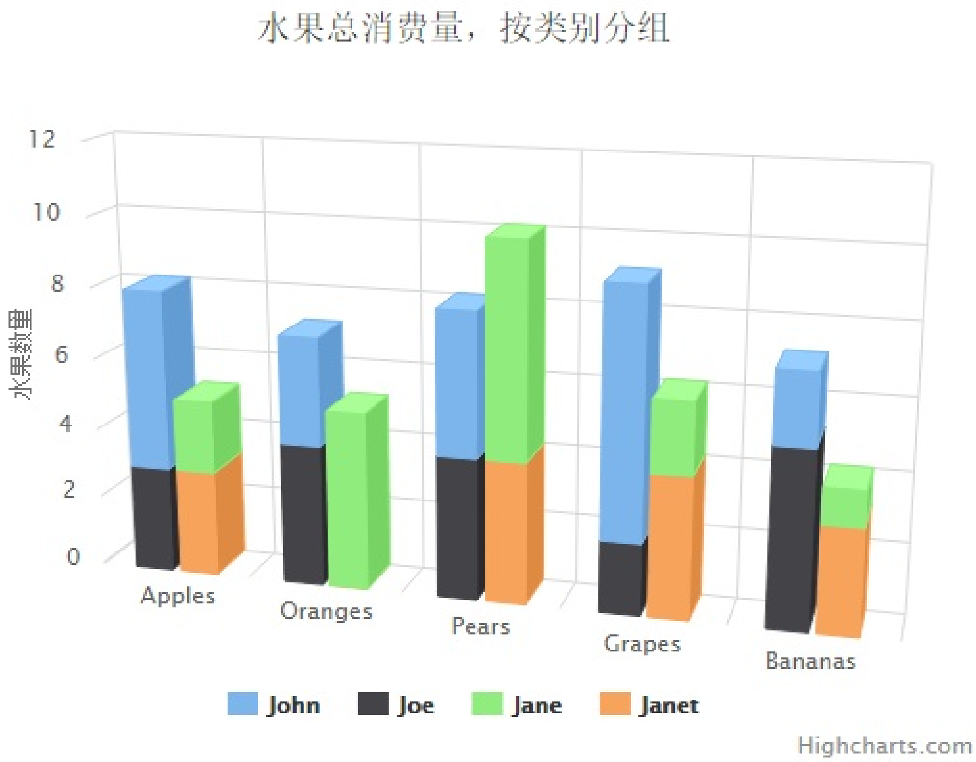
```
var chart = { type: 'column', options3d: { enabled: true, //显
```

实例

文件名：highcharts_3d_stacking.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


水果总消费量，按类别分组
水果数量
12
10
8
6
4
2
0
Apples
Oranges
Pears
Grapes
Bananas
John
Joe
Jane
Janet
Highcharts.com

Highcharts 3D饼图

以下实例演示了3D饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.options3d 配置

以下列出了 3D 图的基本配置，设置 chart 的 type 属性为 pie，options3d 选项可设置三维效果。

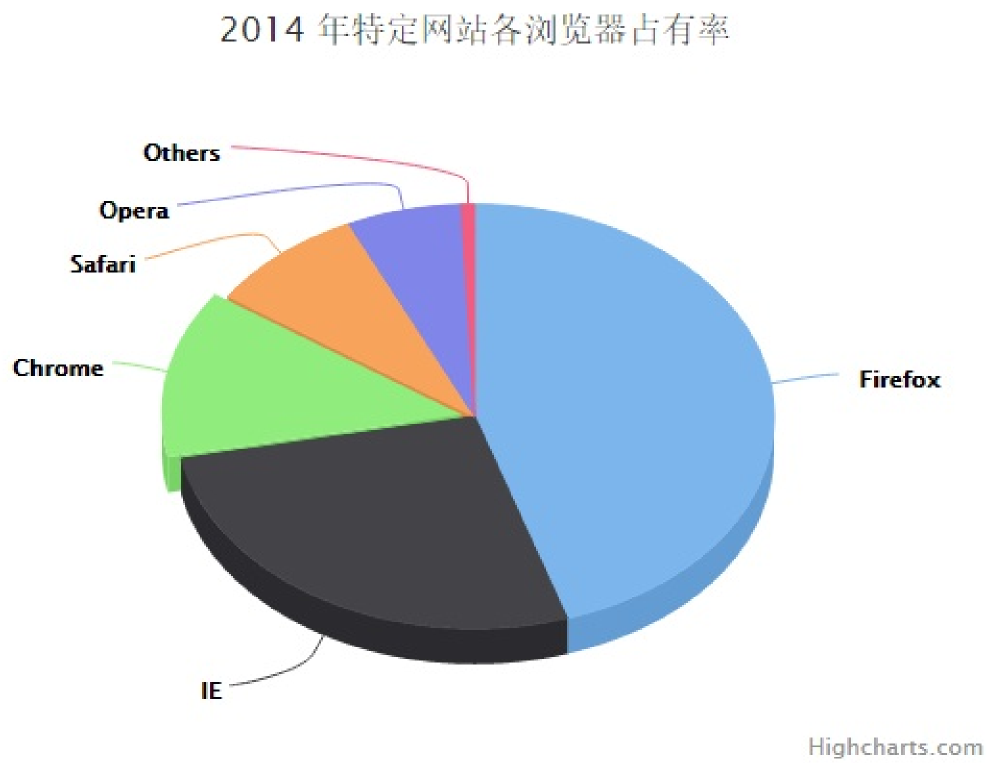
```
var chart = { type: 'pie', options3d: { enabled: true, //显示图
```

实例

文件名：highcharts_3d_pie.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


2014 年特定网站各浏览器占有率
Others
Opera
Safari
Chrome
Firefox
IE
Highcharts.com

Highcharts 3D圆环图

以下实例演示了3D饼图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.options3d 配置

以下列出了 3D 图的基本配置，设置 chart 的 type 属性为 pie，options3d 选项可设置三维效果。

```
var chart = { type: 'pie', options3d: { enabled: true, //显示图
```

plotOptions.pie.innerSize

plotOptions.pie.innerSize 用于绘制饼状图时，饼状图的圆心预留多大的空白。

plotOptions.pie.depth

3D饼图的厚度。

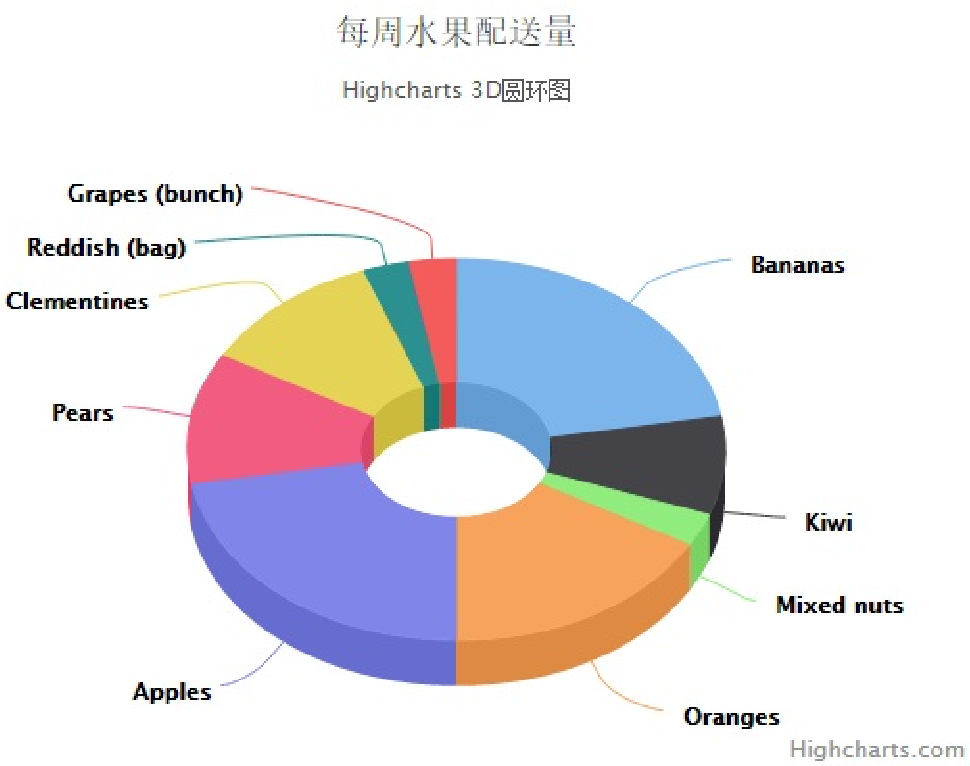
```
plotOptions: { pie: { innerSize: 100, depth: 45 } },
```

实例

文件名：highcharts_3d_donut.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


每周水果配送量
Highcharts 3D圆环图
Grapes (bunch)
Reddish (bag)
Clementines
Pears
Bananas
Kiwi
Mixed nuts
Apples
Oranges
Highcharts.com

Highcharts 测量图

Highcharts 测量图

以下实例演示了测量图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.type 配置

配置 chart 的 type 为 'gauge' 。chart.type 描述了图表类型。默认值为 "line"。

```
var chart = { type: 'guage' };
```

pane 配置

pane 只适用在极坐标图和角度测量仪。此可配置对象持有组合x轴和y周的常规选项。每个x轴和y轴都可以通过索引关联到窗格中。

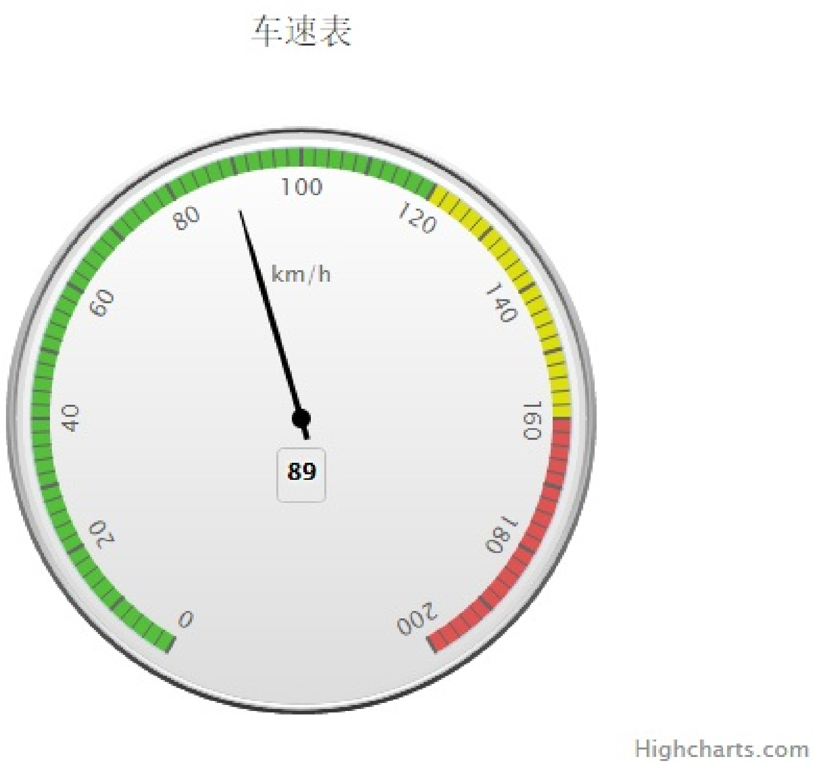
```
var pane = { startAngle: -150, // x轴或测量轴的开始度数，以度数的方式
```

实例

文件名：highcharts_guage_angular.htm

```
<html> <head> <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


车速表
100
80
120
km/h
60
140
40
160
89
20
180
0
200
Highcharts.com

Highcharts 圆形进度条式测量图

以下实例演示了圆形进度条式测量图。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.type 配置

配置 chart 的 type 为 'solidguage' 。chart.type 描述了图表类型。默认值为 "line"。

```
var chart =  { type:  'solidguage'  };
```

pane 配置

pane 只适用在极坐标图和角度测量仪。此可配置对象持有组合x轴和y周的常规选项。每个x轴和y轴都可以通过索引关联到窗格中。

```
var pane =  { startAngle:  -150,  // x轴或测量轴的开始度数，以度数的方式
```

实例

文件名：highcharts_guage_solid.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

Highcharts 时钟

以下实例演示了时钟。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.type 配置

配置 chart 的 type 为 'gauge' 。chart.type 描述了图表类型。默认值为 "line"。

```
var chart =  { type:  'gauge'  };
```

pane 配置

pane 只适用在极坐标图和角度测量仪。此可配置对象持有组合x轴和y周的常规选项。每个x轴和y轴都可以通过索引关联到窗格中。

```
var pane =  { startAngle:  -150,  // x轴或测量轴的开始度数，以度数的方式
```

实例

文件名：highcharts_guage_clock.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
   * 获取当前时间
   */  function getNow()  {  var now =  new  Date();  return  { hou
   * Pad numbers
   */  function pad(number, length)  {  // Create an array of the 
```

以上实例输出结果为：

Highcharts 时钟



Highcharts 双轴车速表

以下实例演示了双轴车速表。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.type 配置

配置 chart 的 type 为 'gauge' 。chart.type 描述了图表类型。默认值为 "line"。

```
var chart =  { type:  'gauge'  };
```

pane 配置

pane 只适用在极坐标图和角度测量仪。此可配置对象持有组合x轴和y周的常规选项。每个x轴和y轴都可以通过索引关联到窗格中。

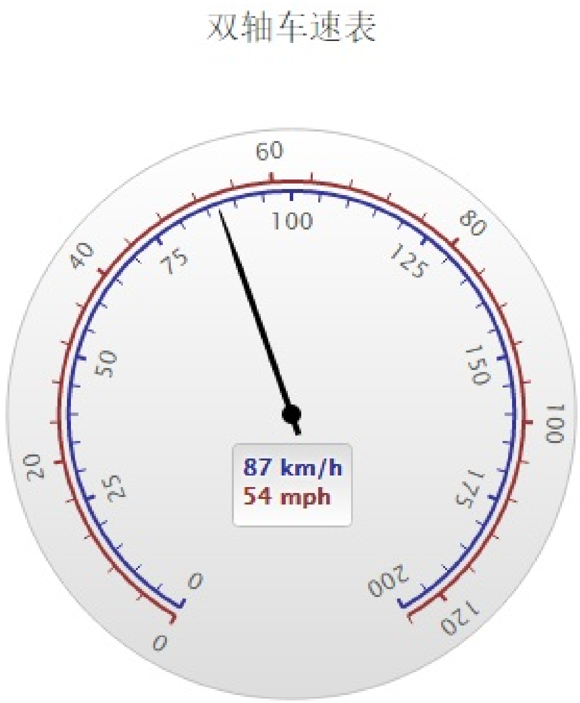
```
var pane =  { startAngle:  -150,  // x轴或测量轴的开始度数，以度数的方式
```

实例

文件名：highcharts_guage_dualaxes.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


双轴车速表
60
100
80
40
75
125
50
150
100
20
25
175
0
200
0
120
87 km/h
54 mph

Highcharts 音量表（VU Meter）

以下实例演示了音量表（VU Meter）。

我们在前面的章节已经了解了 Highcharts 基本配置语法。接下来让我们来看下其他的配置。

配置

chart.type 配置

配置 chart 的 type 为 'gauge' 。chart.type 描述了图表类型。默认值为 "line"。

```
var chart =  { type:  'gauge'  };
```

pane 配置

pane 只适用在极坐标图和角度测量仪。此可配置对象持有组合x轴和y周的常规选项。每个x轴和y轴都可以通过索引关联到窗格中。

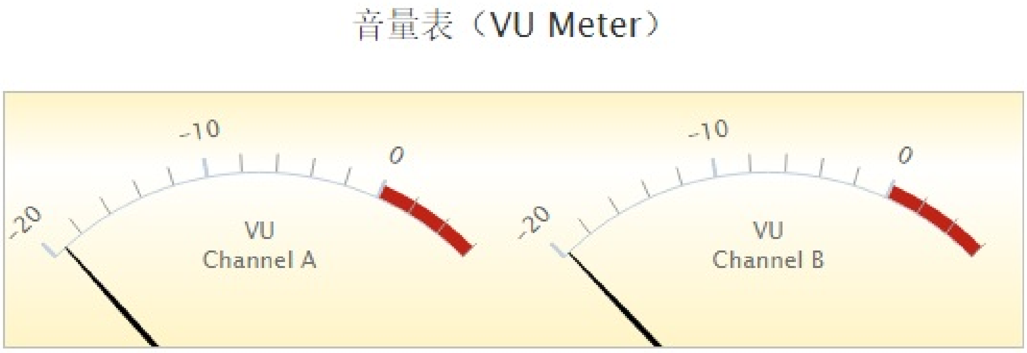
```
var pane =  { startAngle:  -150,  // x轴或测量轴的开始度数，以度数的方式
```

实例

文件名：highcharts_vumeter.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

以上实例输出结果为：


音量表（VU Meter）
-20
-10
0
VU
Channel A
-20
-10
0
VU
Channel B

Highcharts 树状图(Treemap)

本章节我们将为大家介绍 Highcharts 的热点图。

我们在前面的章节已经了解了 Highcharts 配置语法。接下来让我们来看下 Highcharts 的其他配置。

树状图

series 配置

设置 series 的 type 属性为 treemap ， series.type 描述了数据列类型。默认值为 "line"。

```
var chart =  { type:  'treemap'  };
```

实例

文件名：highcharts_tree_map.htm

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

不同等级树状图

以下实例使用不同颜色来标识不同等级的树状图。

实例

文件名：highcharts_tree_levels.htm（完整源码请点击实例查看）

```
<html>  <head>  <title>Highcharts 教程 | 菜鸟教程(runoob.com)</title>
```

大数据量树状图

以下实例颜色了大数据量的树状图，具体实例数据可通过点击"尝试一下"查看。

文件名：highcharts_tree_largemap.htm

```
<html>  <head>  <title>Highcharts Tutorial</title>  <script  src="h
```

Google 地图API

Google 地图API Key

开始学习本教程前，你需要拥有一个免费的 Google 地图 API key。

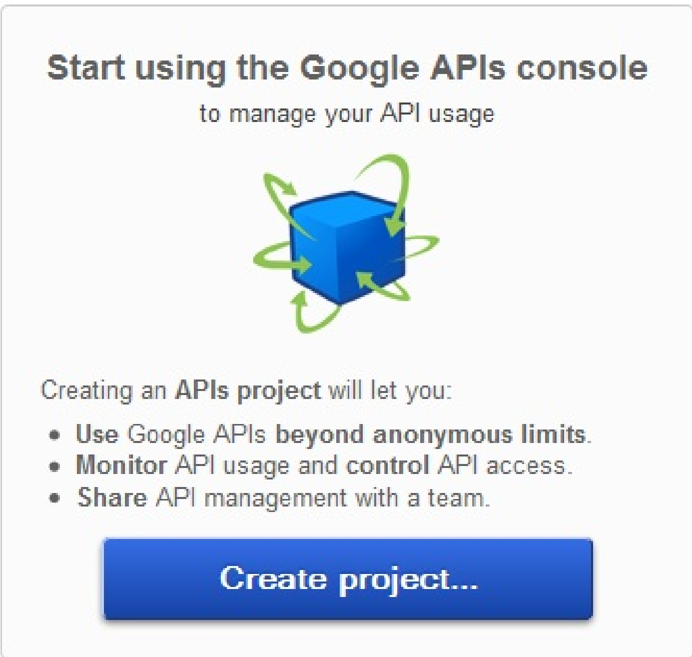
开始学习？

开始学习本教程前，你需要在Google上申请一个指定的API key。

通过以下步骤我们可以免费获取 API key 。

访问 https://code.google.com/apis/console/, 使用你的Google账号登陆。

登陆后会出现如下界面：



点击 "Create Project" 按钮。

在服务列表中找到 **Google Maps API v3,** 然后点击 "off"（关闭） 让其开启该服务器

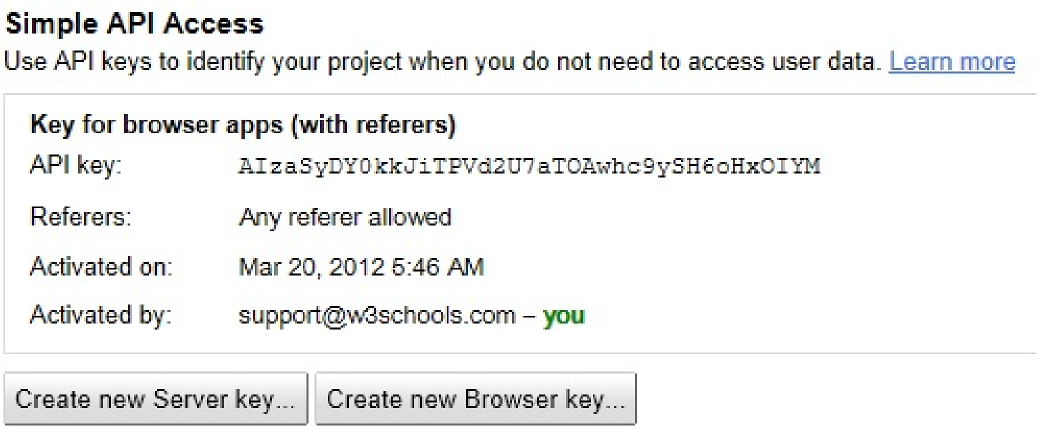
在下一个步骤中，选择"I Agree..." 然后点击 "Accept" 按钮。 现在你在服务列表中应该就可以看到 Google Maps API v3 已经变为 "on"（开启）状态。

接着在左侧菜单中点击 "API Access"， 在右侧栏中将看到以下提示 "Create an OAuth 2.0 client id..."。

点击 "Create an OAuth 2.0 client id..."，将弹出一个表单，表单需要你填入你的项目名称， 项目图片或者logo，然后点击 "Next" 按钮。

然后， 我们需要选择应用类型 ("Web application" ：网站应用), 然后填写你的站点地址， 之后点击 "Create Client Id" 按钮即可。

最后我们就可以得到我们需要的 API key,如下图所示：



注意： 保存你的API key！（在填写的指定 URL 中开发所有的 Google 地图应用你需要使用该API key)。

Google Maps 基础

创建一个简单的 Google 地图

现在让我们创建一个简单的 Google 地图。

以下是显示了英国伦敦的 Google 地图：

实例

```
<!DOCTYPE html>
<html>
<head>
<script src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kk
</script>

<script>
function initialize()
{
var mapProp = {
  center:new google.maps.LatLng(51.508742,-0.120850),
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };
var map=new google.maps.Map(document.getElementById("googleMap")
  ,mapProp);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>

</body>
</html>
```

实例解析

我们以以上实例来解析 Google 地图的创建过程。

应用为什么要声明 HTML5?

```
<!DOCTYPE html>
```

大多数浏览器使用 "标准模式" 的 HTML5 文档渲染页面，这就意味着你的应用是兼容各大浏览器的。

另外，如果没有DOCTYPE标签，浏览器则使用混杂模式 (quirks mode)进行渲染页面内容。

提示： 应该注意的是一些"混杂模式 "中的CSS并不能使用与标准模式中。在具体的应用中，所有基于百分比的大小都必须从父块元素继承 。如果在父模块中没有指定大小，默认值为 0 x 0 像素。如果你想使用百分比，可以在<style> 标签中声明，如下所示：

```
<style type="text/css">
html {height:100%}
body {height:100%;margin:0;padding:0}
#googleMap {height:100%}
</style>
```

这个样式声明表明地图模块的（GoogleMap）应 HTML高度为100%。

添加 Google 地图 API Key

在以下实例中第一个<script> 标签中必须包含 Google 地图 API：

```
<script src="http://maps.googleapis.com/maps/api/js?key=_**YOUR_API
```

将google生成的 API key 放置于 **key** 参数中(key=*YOUR_API_KEY)*。

The **sensor** 参数是必须的，该参数用于指明应用程序是否使用一个传感器 (类似GPS 导航) 来定位用户的位置。参数值可以设置为 true 或者 false。

HTTPS

如果你的应用是安全的HTTP(HTTPS:HTTP Secure)应用,你可以使用 HTTPS 来加载 Google 地图 API：

```
<script src="https://maps.googleapis.com/maps/api/js?key=_**YOUR_AP
```

异步加载

同样我们也可以在页面完全载入后再加载 Google 地图 API。

以下实例使用了 window.onload 来实现页面完全载入后加载 Google 地图 。 loadScript() 函数创建了加载 Google 地图 API <script> 标签。此外在标签的末尾添加了 callback=initialize 参数， initialize()作为回调函数会在API完全载入后执行：

实例

```
function loadScript()
{
var script = document.createElement("script");
script.src = "http://maps.googleapis.com/maps/api/js?      key=AIza
}

window.onload = loadScript;
```

定义地图属性

在初始化地图前，我们需要先创建一个 Map 属性对象来定义一些地图的属性：

```
var mapProp = {
  center:new google.maps.LatLng(51.508742,-0.120850),
  zoom:7,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

center（中心点）

中心属性指定了地图的中心，该中心通过坐标（纬度，经度）在地图上创建一个中心点。

Zoom（缩放级数）

zoom 属性指定了地图的 缩放级数。zoom: 0 显示了整个地球地图的完全缩放。

MapTypeId（地图的初始类型）

mapTypeId 属性指定了地图的初始类型。

mapTypeId包括如下四种类型：

- google.maps.MapTypeId.HYBRID：显示卫星图像的主要街道透明层
- google.maps.MapTypeId.ROADMAP：显示普通的街道地图
- google.maps.MapTypeId.SATELLITE：显示卫星图像
- google.maps.MapTypeId.TERRAIN：显示带有自然特征（如地形和植被）的地图

在哪里显示 Google 地图

通常 Google 地图使用于 <div> 元素中：

```
<div id="googleMap" style="width:500px;height:380px;"></div>
```

注意： 地图将以div中设置的大小来显示地图的大小，所以我们可以在 <div> 元素中设置地图的大小。

创建一个 Map 对象

```
var map=new google.maps.Map(document.getElementById("googleMap")
,mapProp);
```

以上代码使用参数(mapProp)在<div> 元素 (id为googleMap) 创建了一个新的地图。

提示：如果想在页面中创建多个地图，你只需要添加新的地图对象即可。

以下实例定义了四个地图实例 (四个地图使用了不同的地图类型):

实例

```
var map = new google.maps.Map(document.getElementById("googleMap"),
var map2 = new google.maps.Map(document.getElementById("googleMap2'
var map3 = new google.maps.Map(document.getElementById("googleMap3'
var map4 = new google.maps.Map(document.getElementById("googleMap4'
```

加载地图

窗口载入后通过执行 initialize() 函数来初始化 Map 对象，这样可以确保在页面完全载入后再加载 Google 地图：

```
google.maps.event.addDomListener(window, 'load', initialize);
```

Google 地图叠加层

在Google地图中添加一个标记

Google 地图 - 叠加层

叠加层是地图上绑定到经度/纬度坐标的对象，会随您拖动或缩放地图而移动。

Google 地图 API 有如下几种叠加层：

- 地图上的点使用标记来显示，通常显示自定义图标。标记是 GMarker 类型的对象，并且可以利用 GIcon 类型的对象来自定义图标。
- 地图上的线使用折线（表示点的集合）来显示。线是类型为 GPolyline 的对象。
- 地图上的区域显示为多边形（如果是任意形状的区域）或底面叠加层（如果是矩形区域）。多边形类似于闭合的折线，因此可以是任何形状。地面叠加层通常用于地图上与图块有直接或间接关联的区域。
- 地图本身使用图块叠加层显示。如果您有自己的系列图块，可以使用 GTileLayerOverlay 类来改变地图上已有的图块，甚至可以使用 GMapType 来创建您自己的地图类型。
- 信息窗口也是一种特殊的叠加层。但是请注意，信息窗口会自动添加到地图中，并且地图只能添加一个类型为 GInfoWindow 的对象。

Google 地图 - 添加标记

记标识地图上的点。默认情况下，它们使用 G_DEFAULT_ICON（您也可以指定自定义图标）。GMarker 构造函数将 GLatLng 和 GMarkerOptions（可选）对象作为参数。

标记设计为可交互。例如，默认情况下它们接收 "click" 事件，常用于在事件侦听器中打开信息窗口。

通过 setMap() 方法在地图上添加标记:

实例

```
var marker=new google.maps.Marker({
  position:myCenter,
  });

marker.setMap(map);
```

Google 地图 - 可拖动的标记

以下实例将减少如何使用 animation 属性来拖动标记:

实例

```
marker=new google.maps.Marker({
  position:myCenter,
  animation:google.maps.Animation.BOUNCE
  });

marker.setMap(map);
```

Google 地图 - 图标

标记可以用自定义的新图标来显示，以替代默认图标:

实例

```
var marker=new google.maps.Marker({
  position:myCenter,
  icon:'pinkball.png'
  });

marker.setMap(map);
```

Google 地图 - 折线

GPolyline 对象可在地图上创建线性叠加层。GPolyline 包括一系列点，并创建一系列有序连接这些点的线段。

折线支持以下属性：

- path - 指定了多个直线的纬度/经度坐标
- strokeColor - 指定直线的十六进制颜色值(格式: "#FFFFFF")
- strokeOpacity - 指定直线的透明度(该值为 0.0 到 1.0)
- strokeWeight - 定义线的宽度,以像素为单位。
- editable - 定义用户是否可编辑直线(true/false)

实例

```
var myTrip = [stavanger,amsterdam,london];
var flightPath = new google.maps.Polyline({
  path:myTrip,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2
});
```

Google 地图 - 多边形

GPolygon 对象类似于 GPolyline 对象，因为它们都包括一系列有序的点。但是，多边形不像折线一样有两个端点，而是设计为定义形成闭环的区域。

和折线一样，您可以自定义多边形边（线）的颜色、粗细和透明度，以及封闭的填充区域的颜色和透明度。颜色应是十六进制数字 HTML 样式。

多边形支持以下属性：

- path - 指定多个直线纬度的坐标 (第一个和最后一个坐标是相等的)
- strokeColor - 指定直线的十六进制颜色值(格式: "#FFFFFF")
- strokeOpacity -指定直线的透明度(该值为 0.0 到 1.0)
- strokeWeight - 定义线的宽度,以像素为单位。
- fillColor - 指定闭合区域的十六进制颜色值 (格式: "#FFFFFF")
- fillOpacity - 指定填充颜色的透明度 (该值为 0.0 到 1.0)
- editable - 定义用户是否可编辑直线(true/false)

实例

```
var myTrip = [stavanger,amsterdam,london,stavanger];
var flightPath = new google.maps.Polygon({
  path:myTrip,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2,
  fillColor:"#0000FF",
  fillOpacity:0.4
});
```

Google 地图 - 圆

圆支持以下属性：

- center - 指定圆的中心点参数 google.maps.LatLng
- radius - 指定圆的半径，以米为单位
- strokeColor - 指定弧线的十六进制颜色值(格式: "#FFFFFF")

- strokeOpacity - 指定弧线的透明度(该值为 0.0 到 1.0)
- strokeWeight -定义线的宽度,以像素为单位。
- fillColor - 指定圆的十六进制颜色值填充值 (格式: "#FFFFFF")
- fillOpacity - 指定填充颜色的透明度 (该值为 0.0 到 1.0)
- 定义用户是否可编辑直线(true/false)

实例

```
var myCity = new google.maps.Circle({
  center:amsterdam,
  radius:20000,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2,
  fillColor:"#0000FF",
  fillOpacity:0.4
});
```

Google 地图 - 信息窗口

在一个标记上显示一个文本信息窗口：

实例

```
var infowindow = new google.maps.InfoWindow({
  content:"Hello World!"
  });

infowindow.open(map,marker);
```

Google 地图 - 叠加层参考手册

Google Maps API 参考手册.

Google 地图事件

点击标记缩放地图 - 绑定在google地图上的事件。

点击标记缩放地图

我们仍然使用上一遍文章使用的英国伦敦的地图。

点用户点击标记时实现缩放地图的功能(点击标记时绑定地图缩放事件)。

代码如下：

实例

```
// Zoom to 9 when clicking on marker
google.maps.event.addListener(marker,'click',function() {
  map.setZoom(9);
  map.setCenter(marker.getPosition());
  });
```

使用 addListener() 事件处理程序来注册事件的监听。该方法使用一个对象，一个事件来监听，当指定的事件发生时 函数将被调用。

重置标记

我们通过给地图添加事件处理程序来改变 'center' 属性，以下代码使用 center_changed 事件在3秒后标记移会中心点:

实例

```
google.maps.event.addListener(map,'center_changed',function() {
  window.setTimeout(function() {
    map.panTo(marker.getPosition());
  },3000);
});
```

点击标记时打开信息窗口。

点击标记在信息窗口显示一些文本信息：

实例

```
var infowindow = new google.maps.InfoWindow({
  content:"Hello World!"
  });

google.maps.event.addListener(marker, 'click', function() {
  infowindow.open(map,marker);
  });
```

设置标记及打开每个标记的信息窗口

当用户点击地图时执行一个窗口

用户点击地图某个位置时使用 placeMarker() 函数在指定位置上放置一个标记，并弹出信息窗口：

实例

```
google.maps.event.addListener(map, 'click', function(event) {
  placeMarker(event.latLng);
  });

function placeMarker(location) {
  var marker = new google.maps.Marker({
    position: location,
    map: map,
  });
  var infowindow = new google.maps.InfoWindow({
    content: 'Latitude: ' + location.lat() +
    '<br>Longitude: ' + location.lng()
  });
  infowindow.open(map,marker);
}
```

Google 地图 - 事件参考手册

Google Maps API 参考手册。

Google 地图控件集

一个Google 地图 - 默认控件集设置：

Google 地图 - 默认控件集设置：

当使用一个标准的google地图，它的默认设置如下：

- .Zoom-显示一个滑动条来控制map的Zoom级别
- .PPan-地图上显示的是一个平底碗样的控件，点击4个角平移地图
- .MapType-允许用户在map types(roadmap 和 satallite)之间切换
- .StreetView-显示为一个街景小人图标，可拖拽到地图上某个点来打开街景

Google 地图 - 更多控件集

除了以上默认控件集,Google还有：

- .Scale-显示地图比例尺
- .Rotate-显示一个小的圆周图标，可以转动地图
- .verview Map-从一个广域的视角俯视地图

创建地图时你可以通过设置选项指定哪些控件集显示或者通过调用 setOptions() 来改变地图的设置选项。

Google 地图 - 关闭默认控件集

你可能希望能够关闭默认的控件集。

为了关闭默认控件集,设置地图的disableDefaultUI的属性为true:

实例

```
disableDefaultUI:true
```

Google 地图 - 开所有控件集

一些控件集默认显示在地图上，而其它的不会，除非你设置它们。

设置控件为true使其可见-设置控件为false则隐藏它。

以下实例开启所有的控件：

实例

```
panControl:true,
zoomControl:true,
mapTypeControl:true,
scaleControl:true,
streetViewControl:true,
overviewMapControl:true,
rotateControl:true
```

Google 地图 - 修改控件集

某些地图控件是可配置的。通过制定控件选项域改变控件集。

F举个例子来说，修改Zoom 控件的选项在zoomControlOptions指定。zoomControlOptions包含如下3种选项：

- .google.maps.ZoomControlStyle.SMALL-显示最小化zoom 控件
- .google.maps.ZoomControlStyle.LARGE-显示标准zoom滑动控件
- .google.maps.ZoomControlStyle.DEFAULT-基于设备和地图大小，选择最合适的控件

实例

```
zoomControl:true,
zoomControlOptions:  {
  style:google.maps.ZoomControlStyle.SMALL
}
```

注意： 如果需要修改一个控件，首先开启控件(设置其为true)。

另一个控件为 MapType 控件。

MapType 控件可显示为以下 style 选项之一：

- google.maps.MapTypeControlStyle.HORIZONTAL_BAR，用于在水平栏中将一组控件显示为如 Google Maps 中所示按钮。
- google.maps.MapTypeControlStyle.DROPDOWN_MENU，用于显示单个按钮控件，以便您从下拉菜单中选择地图类型。
- google.maps.MapTypeControlStyle.DEFAULT，用于显示"默认"的行为，该行为取决于屏幕尺寸，并且可能会在 API 以后的版本中有所变化。

实例

```
mapTypeControl:true,
mapTypeControlOptions: {
  style:google.maps.MapTypeControlStyle.DROPDOWN_MENU
}
```

同样你可以使用ControlPosition属性指定控件的位置:

实例

```
mapTypeControl:true,
mapTypeControlOptions: {
  style:google.maps.MapTypeControlStyle.DROPDOWN_MENU,
  position:google.maps.ControlPosition.TOP_CENTER
}
```

Google 地图 - 自定义控件集

创建一个返回伦敦自定义控件，用于点击事件： (如果地图被拖拽):

实例

```
controlDiv.style.padding = '5px';
var controlUI = document.createElement('div');
controlUI.style.backgroundColor = 'yellow';
controlUI.style.border='1px solid';
controlUI.style.cursor = 'pointer';
controlUI.style.textAlign = 'center';
controlUI.title = 'Set map to London';
controlDiv.appendChild(controlUI);
var controlText = document.createElement('div');
controlText.style.fontFamily='Arial,sans-serif';
controlText.style.fontSize='12px';
controlText.style.paddingLeft = '4px';
controlText.style.paddingRight = '4px';
controlText.innerHTML = '<b>Home<b>'
controlUI.appendChild(controlText);
```

Google 地图 - 控件集参考手册

Google Maps API 参考手册.

Google 地图类型

HYBRID类型的google地图:

Google 地图- 基本地图类型

Google Maps API 中提供了以下地图类型：

- `MapTypeId.ROADMAP`，用于显示默认的道路地图视图
- `MapTypeId.SATELLITE`，用于显示 Google 地球卫星图片
- `MapTypeId.HYBRID`，用于同时显示普通视图和卫星视图
- `MapTypeId.TERRAIN`，用于根据地形信息显示实际地图。

要通过 Map 修改正在使用的地图类型，您可以为其设置 mapTypeId 属性:

```
var mapProp = {
  center:new google.maps.LatLng(51.508742,-0.120850),
  zoom:7,
  mapTypeId: google.maps.MapTypeId.HYBRID
};
```

或者动态修改 mapTypeId：

```
map.setMapTypeId(google.maps.MapTypeId.HYBRID);
```

Google 地图- 45° 图像

Google Maps API 针对特定位置支持特殊的 45° 图像。

此类高分辨率图像可提供朝向各基本方向（东南西北）的透视视图。对于支持的地图类型，这些图片还可提供更高的缩放级别。

现有的 google.maps.MapTypeId.SATELLITE 和 google.maps.MapTypeId.HYBRID 地图类型支持高缩放级别的 45° 透视图像（如果有的话）。如果您放大的位置拥有此类图像，那么这些地图类型将会自动通过以下方式更改其视图：

- 地图上现有的任何平移控件都将会变更为在现有的导航控件周围添加一个罗盘转轮。您可以通过该罗盘来更改任意 45° 图像的方向，方法是：拖动该罗盘转轮，然后将方向对准包含图像的最近支持方向。
- 一个旋转控件将会间隙显示在现有的平移和缩放控件之间，它可用于将图像围绕支持方向旋转。旋转控件仅支持顺时针方向旋转。
- 以当前位置为中心的 45° 透视图像将会替代卫星图像或混合图像。默认情况下，此类视图会朝向北方。如果您缩小地图，则地图会重新显示默认的卫星图

像或混合图像。
- MapType 控件将启用子菜单切换控件，用于显示 45° 图像。

注意：缩小显示 45° 图像的地图类型将会还原所有更改，并重新构建原始地图类型。

以下示例显示了意大利威尼斯公爵宫45°视图：

实例

```
var mapProp = {
center:myCenter,
zoom:18,
mapTypeId:google.maps.MapTypeId.HYBRID
};
```

提示：Google 正在不断地为更多城市添加 45° 图像。有关最新信息，请参阅 Google 地图上的 45° 图像列表。

Google 地图 - 启用和停用 45° 图像 - setTilt(0)

您可以通过在 Map 对象上调用 setTilt(0) 来停用 45° 图像。要启用适用于支持的地图类型的 45° 透视图像，请调用 setTilt(45)。

实例

```
map.setTilt(0);
```

Google 地图 - 参考手册

Google 地图 API 参考手册。

Google 地图 API 参考手册

Google 地图API 参考手册

地图

构造函数/对象	描述
Map()	在指定的 HTML 容器中创建新的地图，该容器通常是一个DIV元素。

叠加层

构造函数/对象	描述
Marker	创建一个标记。
MarkerOptions	标记的选项。由DirectionsRenderer渲染的所有标记都将使用这些选项。
MarkerImage	A structure representing a Marker icon or shadow image
MarkerShape	Defines the marker shape to use in determination of a marker's clickable region (type and coord)
Animation	Specifies animations that can be played on a marker (bounce or drop)
InfoWindow	Creates an info window
InfoWindowOptions	Options for rendering the info window
Polyline	Creates a polyline (contains path and stroke styles)
PolylineOptions	Options for rendering the polyline
Polygon	Creates a polygon (contains path and stroke+fill styles)
PolygonOptions	Options for rendering the polygon
Rectangle	Creates a rectangle (contains bounds and stroke+fill styles)
RectangleOptions	Options for rendering the rectangle
Circle	Creates a circle (contains center+radius and stroke+fill styles)
CircleOptions	Options for rendering the circle
GroundOverlay	
GroundOverlayOptions	
OverlayView	
MapPanes	
MapCanvasProjection	

事件

构造函数/对象	描述
MapsEventListener	It has no methods and no constructor. Its instances are returned from addListener(), addDomListener() and are eventually passed back to removeListener()
event	Adds/Removes/Trigger event listeners
MouseEvent	Returned from various mouse events on the map and overlays

控件集

构造函数/对象	描述
MapTypeControlOptions	Holds options for modifying a control (position and style)
MapTypeControlStyle	Specifies what kind of map control to display (Drop-down menu or buttons)
OverviewMapControlOptions	Options for rendering of the overview map control (opened or collapsed)
PanControlOptions	Options for rendering of the pan control (position)
RotateControlOptions	Options for rendering of the rotate control (position)
ScaleControlOptions	Options for rendering of the scale control (position and style)
ScaleControlStyle	Specifies what kind of scale control to display
StreetViewControlOptions	Options for rendering of the street view pegman control (position)
ZoomControlOptions	Options for rendering of the zoom control (position and style)
ZoomControlStyle	Specifies what kind of zoom control to display (large or small)
ControlPosition	Specifies the placement of controls on the map

地图 API Map() 构造器

实例

创建一个 Google 地图:

```
var map=new google.maps.Map(document.getElementById("googleMap"),ma
```

定义和用法

Map() 构造器创建了一个新的地图并插入到指定的HTML元素中（<div> 元素)。

语法

```
new google.maps.Map(_HTMLElement_,_MapOptions_)
```

参数值

参数	描述
HTMLElement	规定要把地图放置在那个 HTML 元素中。
MapOptions	带有地图初始化变量/选项的 MapOptions 对象。

Map() 的方法

方法	返回值	描述
fitBounds(*LatLngBounds*)	None	设置要包含给定 视口。
getBounds()	*LatLng,LatLng*	返回当前视口的 度/经度和东北纬 度。
getCenter()	*LatLng*	返回地图的中心 经度。

getDiv()	*Node*	返回包含地图的 对象。
getHeading()	*number*	返回航拍图像的 向（支持 SATEI 和 HYBRID 地图 型）。
getMapTypeId()	HYBRID ROADMAP SATELLITE TERRAIN	返回当前地图类
getProjection()	*Projection*	返回当前 Projection（投影
getStreetView()	*StreetViewPanorama*	返回绑定到地图 的 StreetViewPanc
getTilt()	*number*	返回航拍图像的 度数（支持 SATELLITE 和 HYBRID 地图类
getZoom()	*number*	返回地图的当前 别。
panBy(*xnumber,ynumber*)	None	通过以像素计的 离改变地图的中
panTo(*LatLng*)	None	改变地图的中心 的 LatLng。
panToBounds(*LatLngBounds*)	None	将地图平移所需 距离以包含给定 LatLngBounds。
setCenter(*LatLng*)	None	
setHeading(*number*)	None	设置航拍图像的 向（以度为单位 量），基本方向 方。
setMapTypeId(*MapTypeId*)	None	改变要显示的地 型。
setOptions(*MapOptions*)	None	
setStreetView(*StreetViewPanorama*)	None	绑定一个 StreetViewPanc 到地图上。
		设置航拍图像的

setTilt(*number*)	None	度数（支持 SATELLITE 和 HYBRID 地图类
setZoom(*number*)	None	

Map() 的属性

属性	类型	描述
controls	*Array.* <MVCArray.<Node>>	要附加到地图上的额外控件。
mapTypes	*MapTypeRegistry*	按字符串 ID 划分的 MapType 实例的注册表。
overlayMapTypes	*MVCArray.* <MapType>	要叠加的额外地图类型。

Map() 的事件

事件	参数	描述
bounds_changed	None	当可视区域范围更改时会触发此事件。
center_changed	None	当地图 center（中心）属性更改时会触发此事件。
click	*MouseEvent*	当用户点击地图（但不是点击标记或信息窗口）时会触发此事件。
dblclick	*MouseEvent*	当用户双击地图时会触发此事件。请注意，触发此事件前还会触发点击事件。
drag	None	当用户拖动地图时会反复触发此事件。
dragend	None	当用户停止拖动地图时会触发此事件。
dragstart	None	当用户开始拖动地图时会触发此事件。
heading_changed	None	当地图 heading（方向）属性更改时会触发此事件。
idle	None	当地图在平移或缩放之后变为闲置状态时会触发此事件。
maptypeid_changed	None	当 mapTypeId 属性更改时会触发此事件。
mousemove	*MouseEvent*	只要用户的鼠标在地图容器上移动，就会触发此事件。
mouseout	*MouseEvent*	当用户的鼠标从地图容器上退出时会触发此事件。
mouseover	*MouseEvent*	当用户的鼠标进入地图容器时会触发此事件。
projection_changed	None	当投影更改时会触发此事件。
resize	None	当地图（div）更改尺寸时会触发此事件。
rightclick	*MouseEvent*	当用户右击地图时会触发此事件。
tilesloaded	None	当可见图块载入完成后会触发此事件。
tilt_changed	None	当地图 tilt（倾斜）属性更改时会触发此事件。
zoom_changed	None	当地图 zoom（缩放）属性更改时会触发此事件。

免责声明

版权信息

菜鸟教程（www.runoob.com）刊载的所有内容，包括文字、图片、音频、视频、软件、程序、以及网页版式设计等均在网上搜集。

菜鸟教程提供的内容仅用于个人学习、研究或欣赏。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关

访问者可将本网站提供的内容或服务用于个人学习、研究或欣赏，以及其他非商业性或非盈利性用途，但同时应遵守著作权法及其他相关法律的规定，不得侵犯本网站及相关权利人的合法权利。

本网站内容原作者如不愿意在本网站刊登内容，请及时通知本站，予以删除。

本网站的编程技术内容大部分翻译自 W3Schools Online Web Tutorials 与 Tutorialspoint, 内容原作者如不愿意在本网站刊登内容，请及时通知本站，予以删除。

链接到菜鸟教程

任何网站都可以链接到菜鸟教程的任何页面。

如果您需要在对少量内容进行引用，请务必在引用该内容的页面添加指向被引用页面的链接。

保证

菜鸟教程不提供任何形式的保证。所有与使用本站相关的直接风险均由用户承担。菜鸟教程提供的所有代码均为实例，并不对性能、适用性、适销性或/及其他方面提供任何保证。

菜鸟教程的内容可能包含不准确性或错误。菜鸟教程不对本网站及其内容的准确性进行保证。如果您发现本站点及其内容包含错误，请联系我们以便这些错误得到及时的更正：429240967@qq.com

您的行为

当您使用本站点时，说明您已经同意并接受本页面的所有信息。

联系方式

联系邮箱：429240967@qq.com（相关事务请发函至该邮箱）